THESIS

CELLULAR AUTOMATA: AN APPROACH TO WAVE PROPAGATION AND FRACTURE MECHANICS PROBLEMS

by

Selcuk Hosoglu

December 2006

Thesis Advisor: Young W. Kwon

REPORT DOCUMENTATION PAGE		*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2006	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Cellular Automata: An Approach to Wave Propagation and Fracture Mechanics Problems	**5. FUNDING NUMBERS**
6. AUTHOR(S) Selcuk Hosoglu	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The Cellular Automata (CA) method is based on the idea that the macroscopic behavior of a system can be captured by using simple local rules running at a microscopic level. In other words, a system can be modeled by means of simple local rules that govern the behavior of the whole system. In this thesis a local CA rule set is introduced and a methodology is developed to model physical systems that are governed by one and two dimensional wave equations. One dimensional systems are also successfully modeled by using CA and FEM techniques working as coupled, whereas two dimensional systems could only be modeled in an error margin due to the variation of the introduced time scaling factor when external forces are involved. Also, the applicability of the CA method to fracture mechanics problems is investigated.

14. SUBJECT TERMS Cellular Automata, Wave Propagation, Crack Propagation, Cellular Automata and FEM coupling	**15. NUMBER OF PAGES** 81
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

CELLULAR AUTOMATA: AN APPROACH TO WAVE PROPAGATION AND FRACTURE MECHANICS PROBLEMS

Selcuk Hosoglu
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 2000

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2006

Author:	Selcuk Hosoglu
Approved by:	Young W. Kwon Thesis Advisor
	Anthony J. Healey Chairman, Department of Mechanical and Astronautical Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Cellular Automata (CA) method is based on the idea that the macroscopic behavior of a system can be captured by using simple local rules running at a microscopic level. In other words, a system can be modeled by means of simple local rules that govern the behavior of the whole system. In this thesis a local CA rule set is introduced and a methodology is developed to model physical systems that are governed by one and two dimensional wave equations. One dimensional systems are also successfully modeled by using CA and FEM techniques working as coupled, whereas two dimensional systems could only be modeled in an error margin due to the variation of the introduced time scaling factor when external forces are involved. Also, the applicability of the CA method to fracture mechanics problems is investigated.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my thesis advisor Professor Young W. Kwon for his patient guidance and sincerity during the course of this thesis.

I would also like to thank the Turkish Navy for giving me the opportunity to study at the Naval Postgraduate School.

Finally, I would especially like to thank my mother, Fatma Hosoglu, my father, Ahmet Hosoglu, and my sister, Sinem Hosoglu, for their selfless and endless love and support throughout my life.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. DEFINITION

Instead of providing a formal definition of Cellular Automaton (plural: Cellular Automata) (CA), we can build up its definition through an example. Consider a chess board where each square on the board is considered a cell. However, instead of being comprised of only two colors, each cell can be any one of a finite number of distinct colors, but not a combination of these colors. For example, one cell can be red and another white, but no cell can be half red and half white. Further, on this board, time is also discrete. At every time step, the colors of the cells change according to a rule (transition rule) which is a function of the colors of neighboring cells and cells' own color, and every cell changes its color at the same time. The whole board with this rule is called a cellular automaton.

In general, CA cells do not have to be colored, but they must be in one of a finite number of states at any given time step. These states may be represented by colors, integer numbers (0, 1, 2, …), or any finite alphabet. Usually the number of states is small, but in principle any finite number of states is acceptable. The way that the neighboring cells are defined changes from automaton to automaton. One can only use the four cells on the east, west, north and south (von Neumann neighborhood), or one can use eight cells (in addition to east, west, north and south, northeast, southeast, southwest and northwest – Moore neighborhood). One can use even a hexagonal lattice instead of a square lattice. In fact CA does not have to be on a plane, any number of dimensions is allowed [1].

Initially, it is assumed that every cell is in the same state, with the exception of some finite number of cells which are in a different state. This is called the *configuration*.

Let's give an example. The rule we will discuss here was initially proposed by Edward Fredkin in the 1970's [2]. The rule is defined on a two dimensional plane, where each cell is labeled by its position vector $\vec{r} = (i, j)$, where ***i*** and ***j*** are the row and column indices, respectively. A function $\psi_t(\vec{r})$ is associated to the lattice and describes the state of each cell at iteration ***t***. The state can be 1 or 0. The CA rule defines how to compute the state of each cell at iteration ***t+1*** by using the states at iteration ***t***. We start from an initial condition at time ***t = 0*** with a given configuration of the values $\psi_0(\vec{r})$ on the lattice. The state at time ***t = 1*** will be obtained as follows

1. For each cell, compute the sum of states (1 or 0) on the four nearest neighbors (north, south, east and west). The system is supposed to be periodic in both ***i*** and ***j*** directions so that every cell has four neighbors. Using periodic boundaries assures this calculation to be well defined for all sites.
2. If the sum is even, the new state $\psi_1(\vec{r})$ is 0 (white), otherwise, it is 1 (black).

From a mathematical point of view, this CA rule can be expressed by the following relation

$$\psi_{t+1}(i, j) = \psi_t(i+1, j) \oplus \psi_t(i-1, j) \oplus \psi_t(i, j+1) \oplus \psi_t(i, j-1) \qquad (1)$$

where the symbol $\oplus$ stands for the exclusive-or logical operation. This same rule is repeated to find the states at time *t* = 2, 3, 4, … [3].

In Figure 1 we can see that after some number of iterations the rule leads to very complex shapes. This example shows that despite the simplicity of the local rule, the behavior of a CA model can be quite complex [3].

In this example, the rule is identical for each cell and is applied simultaneously to each of them, leading to a synchronous dynamics. This rule is homogeneous, that is it cannot depend explicitly on the cell position $\vec{r}$. However, spatial or even temporal inhomogeneities can be introduced. Boundary cells are typical examples of spatial inhomogeneities where the boundaries are not supposed to be periodic. In the case of a rectangular CA domain, it is obvious that the cells at the edges do not have as many neighbors as the cells inside the domain do, thus resulting in spatial inhomogeneity. Several approaches can be used to mitigate this concern; including, assigning a constant state to edge cells, or assuming periodic boundaries as we did in our example. Similarly, it is possible to switch between two rules, which one of them is valid at even time steps and the other at odd time steps.

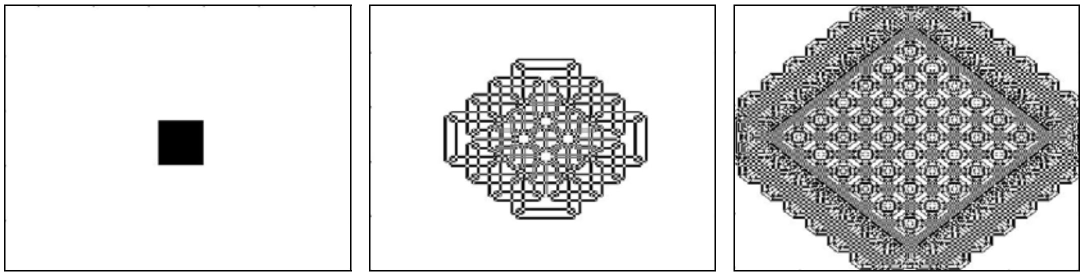


Figure 1. The ⊕ rule on a 256 x 256 periodic lattice. (a) Initial configuration. (b) t = 53 (c) t = 119

According to the rule definition CA is deterministic. The rule is some well defined function and will always evolve identically given the same initial configuration. However, it may be very convenient for some applications to have a certain degree of randomness in the rule. CAs whose updating rule is driven by some external probabilities are called *probabilistic* cellular automata. On the other hand, those which strictly comply with the definition given above are referred to as *deterministic* cellular automata [3].

B. HISTORY AND APPLICABLE AREAS

The history of CA extends back to the 1940's. It was invented by von Neumann, a Hungarian mathematician, to extract the abstract mechanisms leading to self-reproduction of biological organisms [4]. In 1970, British mathematician John Horton Conway invented the "Game of Life", the best known example of a CA [5]. The Game of Life CA is represented on an infinite two dimensional grid of cells, where every cell can have one of two possible states (dead or alive). The rules of this game are

1. Any live cell with less than two neighbors dies
2. Any live cell with more than three neighbors dies
3. Any live cell with two or three neighbors stays unchanged
4. Any dead cell with exactly three neighbor comes to life

The game of life has attracted much interest, because it has shown that very complex patterns can emerge from the application of very simple rules. Another interesting result was that it is always possible to find an initial configuration of cellular space that can reproduce the behavior of any electronic gate, so to imitate any computation process [3].

In the 1980s, studies by S. Wolfram showed that a CA may exhibit many of the behaviors of a continuous system, yet in a much simpler mathematical framework [6, 7]. He further noted that CAs not only behaved in a manner similar to certain dynamical systems, but that they could be used to represent a model of a given physical system. Wolfram also invented the standard naming convention for CA rules, which is based on using the decimal representation of the rule table in binary format [3, 10].

Some of the other related research areas under consideration include fluid dynamics problems like porous media, granular flows, spreading of a liquid droplet and wetting phenomena, microemulsion and physical situations like pattern formation, reaction-diffusion processes, nucleation-aggregation growth phenomena and traffic processes [3].

C. ADVANTAGES AND DISADVANTAGES

The power of the CA approach comes from its simplicity. In modeling a physical system, the traditional methodology (and maybe the only feasible way for a long time) has been to try to solve a set of equations (e.g. a differential equation) that describes all the complex behavior of the system and whose solution gives the desired results. With the increased processing power of computers, a new way has become feasible, rather than trying to model the system as a whole, modeling it as the sum of its parts. By using the CA approach, one can attempt to model the system by means of simple local rules governing the behavior of the whole system. The CA model must use some simple (and intuitive or experimental at some level) local rules at the microscopic level, but at the same time, it must reflect the macroscopic behavior of the physical system under consideration.

Numerically, an advantage of the CA approach is its simplicity and its ease of implementation on computers and parallel machines. In addition, working with Boolean quantities prevents instabilities.

Drawbacks of the CA approach derive mostly from its discrete nature. An important one is the statistical noise requiring a systematic averaging process. Another one is the little flexibility of a rule in order to describe a wider range of physical situations [3]. Wolfram investigated this subject and showed that one can define a universal rule that can emulate the behaviors of most of the other simple rules [10].

According to its definition and its nature, the CA approach seems unsuitable for modeling large-scale moving objects. This is because in the CA approach the only things changing are the states of the cells, not the positions of the cells. To mitigate this shortcoming, a number of suggested models [8, 9] have been developed and will be discussed further in Chapter II.

In 1980s, McNamara and Zanetti [12], Higuares, Jimenes and Succi [13] showed that real number representation of CA cellular states had some advantages over working with Boolean cellular states. This approach is called the Lattice Boltzmann (LB) method and is numerically more efficient than the Boolean dynamics.

Finally, it should be remarked that the CA approach is a methodology of searching for ways to model complex systems in terms of simple local rules that govern the whole. Its richness comes from the microscopic simplicity and rules at the cell level [3, 10].

D. METHODOLOGY

In Chapter II, a rule to model waves and large scale moving objects, suggested by Chopard [6] will be introduced. In Chapters III and IV, the implementation of the rule introduced in Chapter II is investigated on one and two dimensional wave propagation problems, respectively. In Chapter V, the applicability of the CA rule to a fracture mechanics problem is investigated. From this perspective, a methodology to make Finite Element (FE) and CA methods work coupled will also be developed.

II. RULES AND BEHAVIOR OF A CA MODEL

A. THE RULE

Our main interest is in modeling wave propagation problems using the CA approach. Recall, the main elements that constitute a CA domain are cells which can be in a finite number of states, namely, 0, 1, 2, …; white / black; dead / alive; etc. An example of a CA model consisting of cells that can be either white or black (1 or 0) was demonstrated in Chapter I. On an automaton, the particles are not moving from one site to another, only the states of the cells are changing during iterations, without any transport of matter.

Chopard proposed a simple, time-reversible model [8] with features

1. to deal with large-scale objects, which can move and interact with their surroundings
2. to allow these objects to have adjustable mass, energy and momentum
3. to maintain the size and the integrity of these objects during the evolution. Deformations are allowed, but the particles composing them should not spread out in the entire space

In the proposed model, the CA space is composed of particles on a lattice and the springs that connect and hold the particles together on the lattice. Only one particle can be linked to each end of a spring. The particles can be of two kinds, namely white or black particles. The end points of a spring can be either color, and the consecutive particles should be of different colors which means that both neighbors of a particle is of the same color (e.g. both of the neighbors of a white particle must be black). A spring should not fold onto itself, therefore we shall give it an orientation and length. According to this definition, a particle initially on the arbitrary positive side of a spring should not pass to the other side during the evolution process. The particles are allowed to alternate in a three dimensional cubic lattice, but no two particles are allowed to occupy the same lattice position in the same time step (this means that no spring can be zero length or fold onto itself). An example of a one dimensional lattice (***a string***) is illustrated in Figure 2. In this example, the positive x-direction is arbitrarily selected to the right.

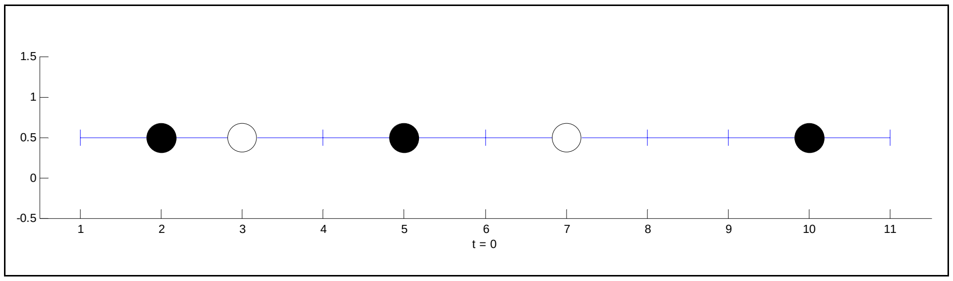


Figure 2. One dimensional CA lattice.

The rule for time evolution of the internal particles (particles with two neighbors) is given as [8]:

$$r(t+1) = r_{+} + r_{-} - r(t) \qquad (2)$$

where $\boldsymbol{r(t)}$ represents the position of a particle at time $\boldsymbol{t}$, and $\boldsymbol{r}_{+}$ and $\boldsymbol{r}_{-}$ represent the positions of two neighboring particles. This equation implies that the position of a particle at the next time step is a reflection of the particle with respect to the center of mass of its two neighbors. This can be demonstrated in Figure 2. Let $\boldsymbol{r(t)}$ represents the position of the second black particle, $\boldsymbol{r(t)=5}$. Therefore, $\boldsymbol{r}_{-}$ and $\boldsymbol{r}_{+}$ represent the positions of first and second white particles, respectively.

It is apparent that Eq.2 is valid for particles with only two neighbors. For the particles at the endpoints, with only one neighbor, the reflection is performed with respect to $\boldsymbol{r}_{\pm} \pm \boldsymbol{a}$, where $\boldsymbol{a}$ is a constant vector which represents the unstretched length and orientation of the spring that links the particles. For convenience a sign convention is used, where the x component of the $\boldsymbol{a}$ vector is assigned a positive value. Also, to prevent the particles from moving off the lattice points, $\boldsymbol{a}$ should be an integer or half integer (assuming that the lattice coordinates are given in integers). We will discuss the $\boldsymbol{a}$ vector in detail in the next section.

The evolution rule for a particle at left end of a string (current position of the particle is ***r(t)***) is given by equation

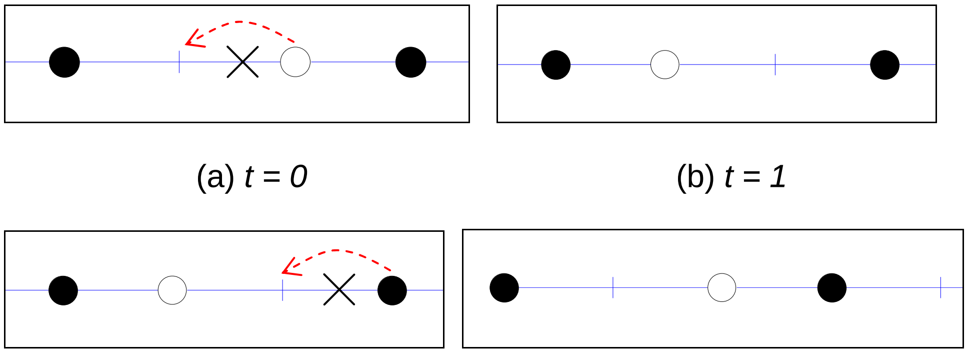
$$r(t+1) = 2(r_+ - a) - r(t) \tag{3}$$

Similarly, the new position of a particle at the right end is

$$r(t+1) = 2(r_- + a) - r(t) \tag{4}$$

The time evolution has two phases. At each time step only one kind of particle can move. For instance, in the first time step, all the white particles are held fixed and the equations described above are applied to black particles only. In the first time step, only black particles change their positions and they move simultaneously. In the next time step, black particles are held fixed and only white particles move on the lattice simultaneously, and so on [8].

Figure 3 demonstrates the time evolution of an internal particle and an end particle on a one dimensional lattice (***a=3/2***). Panel (a) shows the new position of the internal particle which is calculated by reflecting the particle with respect to the center of mass of its two neighboring black particles (Eq.2). Panel (c) shows the new position of the particle at right end of the string which is calculated by reflecting the particle with respect to ($\boldsymbol{r_-} + \boldsymbol{a}$) (Eq.4).



(a) *t = 0* (b) *t = 1*

(c) *t = 1* (d) *t = 2*

Figure 3. Time evolution of one dimensional string particles. (a) and (b) for internal particle, (c) and (d) for right end particle.

In the next section we will investigate the limiting factors of the spring vector ***a*** in detail.

B. SPRING VECTOR

As mentioned in the previous section, there are some constraints in defining the spring vector ***a***. Two of these constraints are that

1. the x component of the ***a*** vector should be positive, and
2. ***a*** should be an integer or half integer to prevent the particles from moving off the lattice points (assuming the lattice coordinates are given as integers).

In addition to these constraints, there are two other requirements on the spring vector. First, a spring should not fold over itself. For instance, a particle at the left end of the string should not pass to the right of its right neighbor in any of the time evolution steps. Similarly, a particle at the right end of the string should not pass to the left of its left neighbor in any of the time evolution steps (right and left are defined assuming that the positive x-coordinate is increasing to the right). Second, the end particles should not go out of the lattice (not the same as moving off the lattice points). This second requirement can be relaxed according to the physical system that is being modeled. If this is the case, one can ignore the constraints on ***a*** that come from this last requirement. We will study these two requirements separately for the left end and right end particles.

1. Left End Particles

The first requirement is that a spring should not fold over itself. This implies that $\boldsymbol{r(t+1)} < \boldsymbol{r}_+$. When we substitute Eq.3 into this, we have

$$\begin{aligned} &r(t+1) < r_+ \\ &2(r_+ - a) - r(t) < r_+ \\ &2r_+ - 2a - r(t) < r_+ \\ &a > \frac{r_+ - r(t)}{2} \end{aligned} \qquad (5)$$

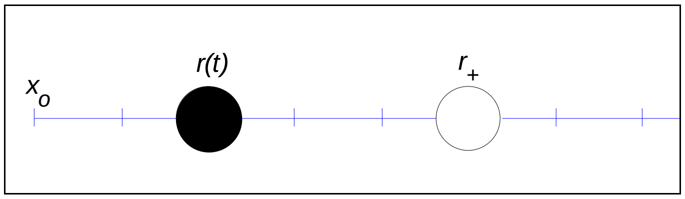


Figure 4. Left end particle.

The last requirement implies that ***r(t+1)* ≥ *x_o***, where $\boldsymbol{x_o}$ represents the left end coordinate of the lattice (Figure 4). Again by substituting Eq.3 into the left hand side of this equation, the following expressions result

$$
\begin{aligned}
&r(t+1) \ge x_o \\
&2(r_+ - a) - r(t) \ge x_o \\
&a \le \frac{x_o + r(t) - 2r_+}{2}
\end{aligned}
\tag{6}
$$

By combining Eq. 5 and 6, we show that the allowed interval of ***a*** for a left end particle on the lattice is

$$
\frac{r_+ - r(t)}{2} < a \le \frac{x_o + r(t) - 2r_+}{2} \tag{7}
$$

2. Right End Particles

Following the same formulation, the first requirement implies that $\boldsymbol{r(t+1) > r_-}$. Substituting Eq.4 into this expression yields

$$
\begin{aligned}
&r(t+1) > r_- \\
&2(r_- + a) - r(t) > r_- \\
&2r_- + 2a - r(t) > r_- \\
&a > \frac{r(t) - r_-}{2}
\end{aligned}
\tag{8}
$$

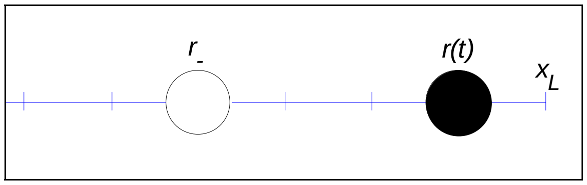


Figure 5. Right end particle.

The last requirement implies that $\boldsymbol{r(t+1) \leq x_L}$, where $\boldsymbol{x_L}$ represents the right end coordinate of the lattice. Substituting Eq.4 into the left hand side of this equation yields

$$\begin{aligned} & r(t+1) \leq x_L \\ & 2(r_- + a) - r(t) \leq x_L \\ & a \leq \frac{x_L + r(t) - 2r_-}{2} \end{aligned} \tag{9}$$

By combining these two, the allowed interval of $\boldsymbol{a}$ for a right end particle on the lattice is

$$\frac{r(t) - r_-}{2} < a \leq \frac{x_L + r(t) - 2r_-}{2} \tag{10}$$

The significance of these results is that if the distance between an end particle and its neighbor is more than $\boldsymbol{2a}$, this rule overshoots the end particle and causes it to move to the other side of its neighbor and fold onto itself. This also implies that the spring changes orientation, which is a violation because we defined the $\boldsymbol{a}$ as a constant. If this distance is exactly $\boldsymbol{2a}$, then the particle collides with its neighbor and tries to occupy the same lattice point with its neighbor. If we are to use this rule, one should bear in mind that a spring linking an end particle and its neighbor cannot stretch more than twice its original length, whereas there is no limit for springs linking internal particles with exactly two neighbors.

C. VELOCITY, MASS, MOMENTUM AND ENERGY

Since the purpose of the proposed time evolution model is to model moving objects, we need to define the velocity of such a model. For this discussion, two kinds of velocities are defined. The first one is the velocity of the string $\boldsymbol{V}$, which is 1 over the required number of time steps to cycle back to its original configuration. The second one is the speed of each particle $\boldsymbol{v_i}$, which is the distance a particle will travel in the next time step (note that every time step is 1 unit time, $\Delta t = 1$).

In Figure 6, it takes 6 time steps for the string to cycle back to its original configuration, and all the particles thus the string moved one lattice point to the right. Therefore, the speed of the string is 1/6.

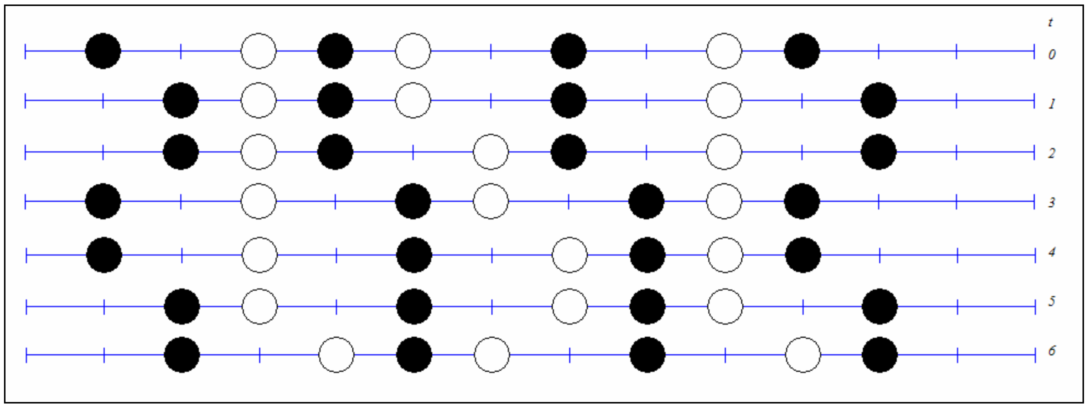


Figure 6. Speed of a string (After [8]).

We define the total number of particles as $\boldsymbol{N}$, and the index $\boldsymbol{i}$ represents the $\boldsymbol{N}$ particles in a string. The indices of the particles increase in the positive x-direction. This convention will assist us in modeling the system in a computer simulation, which is discussed at the end of this chapter.

The total mass of the string is $\boldsymbol{N\text{-}1}$ (1/2 from the end particles and 1 from each of the internal particles). Knowing the velocity and the mass, the momentum of a string is

$$P = \frac{1}{2}v_1 + \sum_{i=2}^{N-1} v_i + \frac{1}{2}v_N \tag{11}$$

In [6] it is shown that the total energy of the string is given by

$$E = \frac{1}{2}\sum_{i=1}^{N-1}\left[(x_{i+1}-x_i-a_x)^2+(y_{i+1}-y_i-a_y)^2+(z_{i+1}-z_i-a_z)^2\right] \tag{12}$$

where $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ represent the spatial coordinates of the particles.

It can be proven that the mass, momentum and energy of a string are conserved during the time evolutions by using the discrete Hamiltonian formalism [8]. According to Eq.11 and Eq.12, one can adjust the mass, momentum and energy of a string by adjusting the number of particles and the initial configuration of the particles on the lattice.

Up to this point we have restricted our examples to one dimensional lattices. Of course this is not always the case. In a multi dimensional lattice, it is important to label / index the particles with increasing numbers towards the positive x-direction (to ensure that we are able to define the left and right neighbors consistently). After that, the time evolution formulation (Eq. 2, 3 and 4) can be applied for all directions separately or one can define all particle locations as vectors and apply the equations as vector algebra. The motions of a string along the $\boldsymbol{x}$ direction is called longitudinal, and along the $\boldsymbol{y}$ and $\boldsymbol{z}$ directions transverse. Figure 7 demonstrates the time evolution of a string on a two dimensional lattice where $\boldsymbol{a=(3/2,0)}$. Note that the internal particles are reflected with respect to the center of mass of their neighbors, and the end particles are reflected with respect to the point which is the vector sum of their neighbor and $\boldsymbol{a}$ vector.

We have already deviated from the classical CA definition by using coordinates of particles rather than the states of cells which allowed us to model a moving object, but in philosophy and methodology we are still using CA approach in the sense that the coordinates of particles in the next time step are still calculated according to a rule that depends on the coordinates of neighboring particles. This rule is spatially and temporally inhomogeneous, because the rules for internal and end particles are different, and they are applied to only one kind of a particle (black / white) at each time step.

We will go one step further and relax the rule, by saying that the particles are no more bounded to lattice points and can go off the lattice points. This will allow us to work with real numbers and bring us a step closer to modeling real physical situations. The practical consequence of this relaxation is that we can ignore the 2nd constraint defined for spring vector $\boldsymbol{a}$ earlier in this section. Therefore, this constant vector is no longer required to be an integer or half integer but can be any real number (of course it must still agree with the other limitations of the spring vector previously discussed).

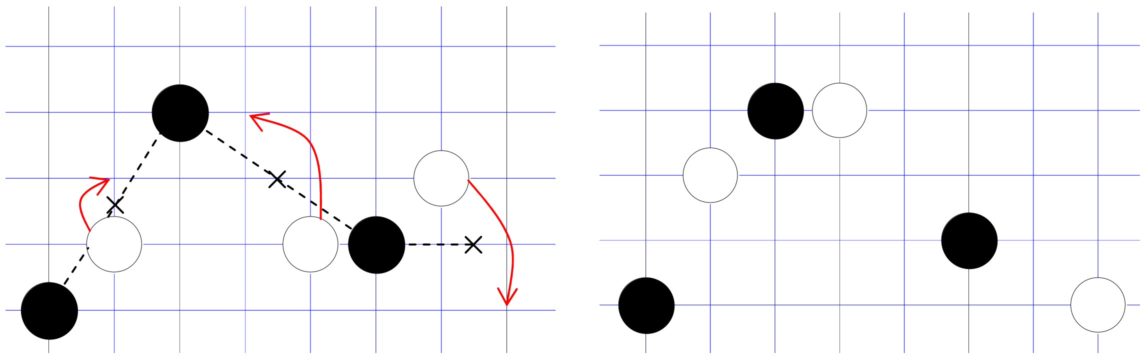


Figure 7. Time evolution of particles on a two dimensional lattice.

D. MODELING CA IN MATLAB

In general, the implementation of CA in a computer program has 7 main steps

1. define the coordinates of black and white particles on the lattice (define the initial configuration of automaton),
2. define spring vector $\boldsymbol{a}$,
3. implement Eq.2, 3 and 4 to black particles,
4. go to next time step,
5. implement Eq.2, 3 and 4 to white particles,
6. go to next time step,
7. go back to step 3

In order to demonstrate this, a MATLAB[1] implementation for a two dimensional CA lattice is developed. First step is to define the coordinates of each particle. In this development, a Cartesian coordinate system is used. The coordinates are stored in an $\boldsymbol{N}$ by 2 matrix, namely $\boldsymbol{r}$, where $\boldsymbol{N}$ is the total number of particles. Every row of this matrix is a 1 by 2 vector that contains the

[1] MATLAB is a registered trademark of MathWorks Inc.

coordinates of each particle. For programming convenience and to make the code simpler, we arbitrarily assume the first particle (left end particle) is always a black particle. This allows us to define the odd indexed rows of the matrix ***r*** as black particles and the even indexed rows to be white particles. Thus, the matrix ***r*** is written

$$r = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ . & . \\ . & . \\ x_N & y_N \end{bmatrix} \begin{array}{ll} \rightarrow & Black \\ \rightarrow & White \\ . & . \\ . & . \\ . & . \end{array}$$

The spring vector is a 1 by 2 vector that defines the orientation and unstretched length of the springs linking the particles.

Steps 3 through 7 are the time evolution steps. In these steps, the rule requires that only white or black particles move in a single time step then the alternate. The choice of which kind of particle to start from is arbitrary, and is determined according to the specific model. Assume we start from the black particles. We know that the coordinates of the black particles are stored in the odd indexed (***i*** = 1, 3, 5 …) rows of matrix ***r***. We construct a loop to apply the rule to black particles. The index ***i*** goes from 1 to N by increments of 2 (***i*** = 1, 3, 5 …), so that we cover all of the black particles. If the index ***i*** is 1 or N, this means that we are dealing with an end particle, and Eq.3 or Eq.4 is used to calculate the next position of ***ith*** particle, otherwise Eq.2 is applied. When the end of the loop is reached, the next time step begins. The same methodology applies to the white particles in the next time step, but this time the index ***i*** goes from 2 to ***N*** by increments of 2 (***i*** = 2, 4, 6 …). The MATLAB code for such an example is given below.

```
%% ********************** START *************************** %%
r=[2,2;
   3,3;
   4,5;                         Step 1
   6,3;
   7,3;
   8,4];
temp=r;
a=[3/2,0];  }  Step 2
t=0;
N=size(r,1);
bt=1; %% Start the time evolution phase with black particles
wt=0;
for t=1:10
    if bt %% Black particles
        for i=1:2:N
            %% Left end particle
            if i==1;
                  temp(i,:)=2*(r(i+1,:)-a)-r(i,:);
                  continue;
            end;
            %% Right end particle                    Step 3
            if i==N;
                  temp(i,:)=2*(r(i-1,:)+a)-r(i,:);
                  continue;
            end;
            %% Internal particle
            temp(i,:)=r(i-1,:)+r(i+1,:)-r(i,:);
        end
    end
    if wt %% White particles
        for i=2:2:N
            %% Left end particle
            if i==1;
                  temp(i,:)=2*(r(i+1,:)-a)-r(i,:);
                  continue;
            end;
            %% Right end particle                    Step 5
            if i==N;                                  Step 7
                  temp(i,:)=2*(r(i-1,:)+a)-r(i,:);
                  continue;
            end;
            %% Internal particle
            temp(i,:)=r(i-1,:)+r(i+1,:)-r(i,:);
        end
    end
    r=temp;
    wt=~wt;
    bt=~bt;
end

%% ********************* END ****************************** %%
```

THIS PAGE INTENTIONALLY LEFT BLANK

III. MODELING ONE DEGREE OF FREEDOM PROBLEMS

A. UNDAMPED SPRING – MASS SYSTEM

The first physical system we try to model using the CA rule set defined in the previous chapter is the undamped free vibration of a spring-mass system. Our primary concern is to determine whether the defined CA rule is able to demonstrate the macroscopic behavior of the real physical system. Therefore, in this first example we will not try to match the analytic solution numerically, but we will investigate the behavior of the CA solution.

A schematic diagram of a notional spring-mass system is shown in Figure 8. The following variables are defined for the system and are enumerated as: spring constant ***k = 100 N/m***, mass ***m = 1 kg***, initial displacement ***x_o = 0.1 m***, initial velocity ***dx(0)/dt = 0*** and unstretched length of the spring ***L = 1 m***. The force acting on the spring from mass ***m*** and the static displacement (δ_s) are calculated as

$$\begin{aligned} F &= mg & F &= k\delta_s \\ F &= 1x9.81 & \delta_s &= 9.81/100 \\ F &= 9.81N & \delta_s &= 0.0981m \end{aligned} \tag{13}$$

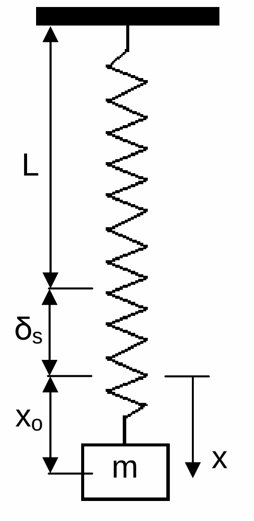


Figure 8. Spring – mass system.

The analytical solution to this problem is given [12] as

$$x(t) = \frac{\dot{x}(0)}{\omega_n}\sin(\omega_n t) + x(0)\cos(\omega_n t)\ ,\ where\ \omega_n = \sqrt{\frac{k}{m}} \tag{14}$$

The CA model consists of 51 particles (26 black and 25 white) which are uniformly spaced on the lattice initially. Since the spring vector $\boldsymbol{a}$ represents the length of each spring in equilibrium, it is calculated as

$$a = \frac{L+\delta_s}{N-1} = \frac{1+0.0981}{21-1} = 0.0549\,m \tag{15}$$

It is important to note that in this case we have a boundary condition, namely the string is fixed at $\boldsymbol{x = 0}$. This boundary condition is implemented in the CA model by ignoring the first particle of the CA. This ensures that no rule is going to be applied to this particle and it will not move, but it will still be used to calculate the time evolution of the second particle as a neighbor. Figure 9 shows a comparison of the CA-derived solution with the analytical solution.

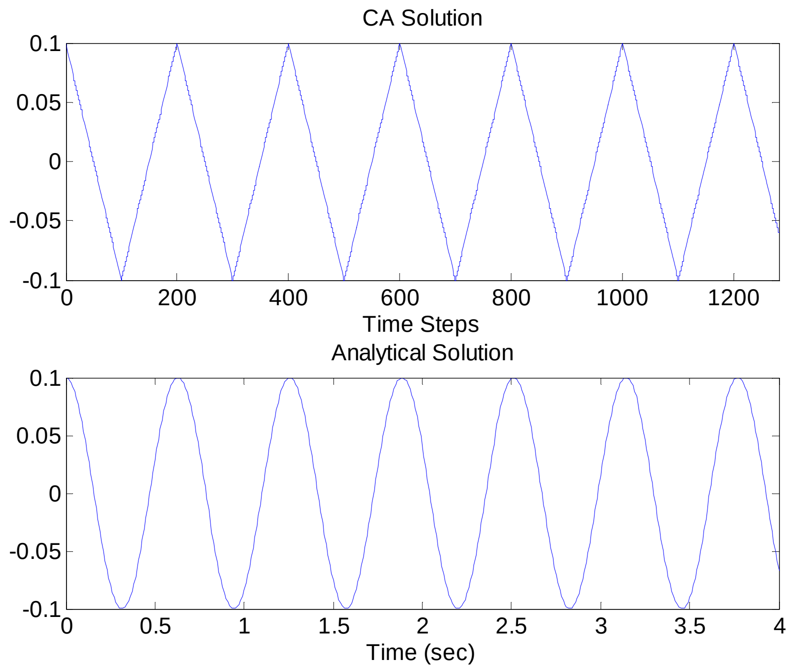


Figure 9. Comparison of CA and analytical solutions of the undamped spring-mass system.

Although not identical, a comparison of these results shows that there is some correlation between the two methodologies. Further, the peaks of the CA solution match the analytical solution and show a simple harmonic behavior like the analytical solution.

This analysis shows that in it is current form the CA rule that is used does not model the undamped spring-mass system perfectly. Further, it is important to note that there is some difference in the time scales; the period of the analytical solution is 0.63 seconds whereas the period of the CA solution is 200 time steps. Therefore, we note that if we use the CA approach to model a real physical system we should also relate the CA time steps to real time. We will discuss and propose a method and introduce the *Time Scale Factor* concept later in this chapter.

B. LONGITUDINAL VIBRATION OF A LONG UNIFORM ROD

The next physical system that we are going to model using the CA approach is the longitudinal vibration of a long uniform rod fixed at one end. In such a system, a force $\boldsymbol{F}$ is initially applied to the rod at its free end and released at $\boldsymbol{t = 0}$. The physical system is represented in Figure 10.

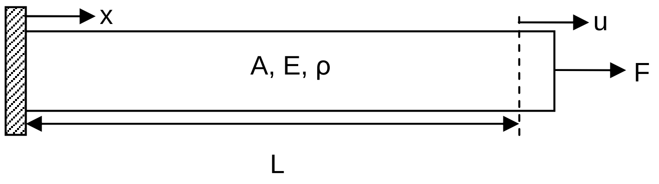


Figure 10. Longitudinal rod.

The analytical solution is given in [13] as

$$u(x,t) = \frac{8FL}{\pi^2 AE} \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^2} \sin\frac{(2n+1)\pi x}{2L} \cos\frac{(2n+1)\pi ct}{2L} \tag{16}$$

We will try to model a steel rod with $\boldsymbol{L = 1\ m}$, $\boldsymbol{A = 10^{-4}\ m^2}$, $\boldsymbol{E = 200\ GPa}$, $\boldsymbol{\rho = 7860\ kg/m^3}$, $\boldsymbol{F = 200\ N}$, by using the CA rule defined in Chapter II. The modeling process is not different from the previous problem. We will use 31 particles (16 black and 15 white particles). The spring vector $\boldsymbol{a}$ is calculated as

$$a = \frac{L}{N-1} = \frac{1}{21-1} = 0.05m \tag{17}$$

At this point if we are to match the temporal part of the analytical solution we must propose a method that relates the discrete time model of CA to real time unit, namely seconds. In a CA model the time steps are originally defined as iterations, so ***Δt = 1*** and it is unitless. We will approach the problem by using the speed of sound definition for solids and introduce a *Time Scaling Factor*. Speed of sound in the beam and speed of sound in the CA model is given as

$$\begin{aligned} c_{real} &= \sqrt{\frac{E}{\rho}} \;;\ \text{speed of sound in beam} \\ \mathrm{c_{CA}} &= \frac{a}{\Delta t} = \frac{a}{1} = \frac{L}{N-1};\ \text{speed of sound in CA model} \end{aligned} \tag{18}$$

Since the distance a sound wave travels in ***t*** seconds should be same in the rod and CA model

$$\mathrm{c_r}\,\Delta t_r = c_{CA}\,\Delta t_{CA} \rightarrow \sqrt{\frac{E}{\rho}}\Delta t_r = \frac{L}{N-1}\Delta t_{CA} \rightarrow \Delta t_r = \Delta t_{CA}\left[\frac{L}{(N-1)\sqrt{\frac{E}{\rho}}}\right] \tag{19}$$

$$Time\,Scaling\ Factor = \left[\frac{L}{(N-1)\,c_r}\right] \quad \rightarrow \Delta t_r = \Delta t_{CA}.TSF$$

When we study the units of this so called *Time Scaling Factor (TSF)*, it should be in time units because ***Δt_{CA}*** is unitless and equals to 1. This can be seen by conducting the following unit analysis.

$$[TSF] = \frac{[L]}{[N-1]\sqrt{\frac{[E]}{[\rho]}}} = \frac{[L]}{\sqrt{\frac{\frac{[M][L]}{[T^2][L^2]}}{\frac{[M]}{[L^3]}}}} = \frac{[L]}{\sqrt{\frac{[L^2]}{[T^2]}}} = \frac{[T][L]}{[L]} = [T] \tag{20}$$

This analysis shows that TSF is in time units and consistent with our definition, so each iteration in a CA model is equal to TSF units in real time.

Concerning our current example, we calculate the initial displacement at the tip of the rod caused by the force $\boldsymbol{F}$, so that we can calculate the initial configuration of the particles on the lattice. The initial displacement is calculated as

$$u(x,0)=\frac{Fx}{AE}\rightarrow u(L,0)=\frac{FL}{AE}=\frac{200x1}{10^{-4}x200x10^{9}}=10^{-5}m \qquad (21)$$

so at $\boldsymbol{t=0}$ the total length of the rod is 1.00001 m. When we compare the analytical and CA solutions at the rod tip and in the middle of the rod, we can see that the CA solution is in perfect agreement with the analytical solution as depicted in Figure 11 and Figure 12.

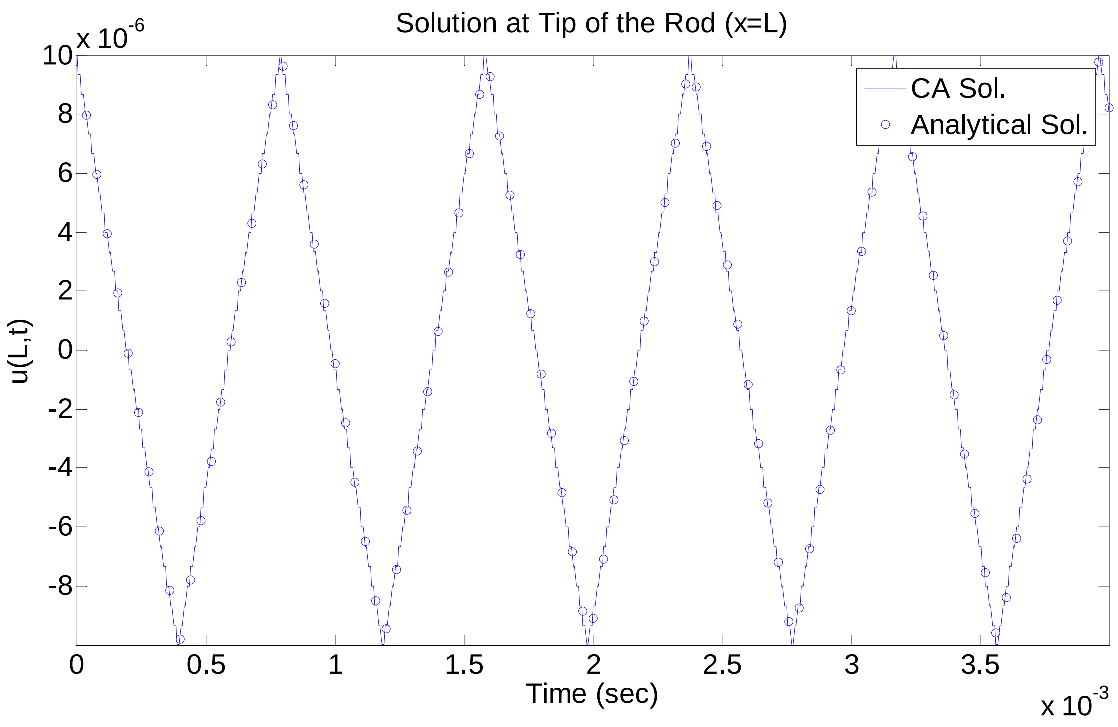


Figure 11. CA and analytical solution of rod problem at x=L.

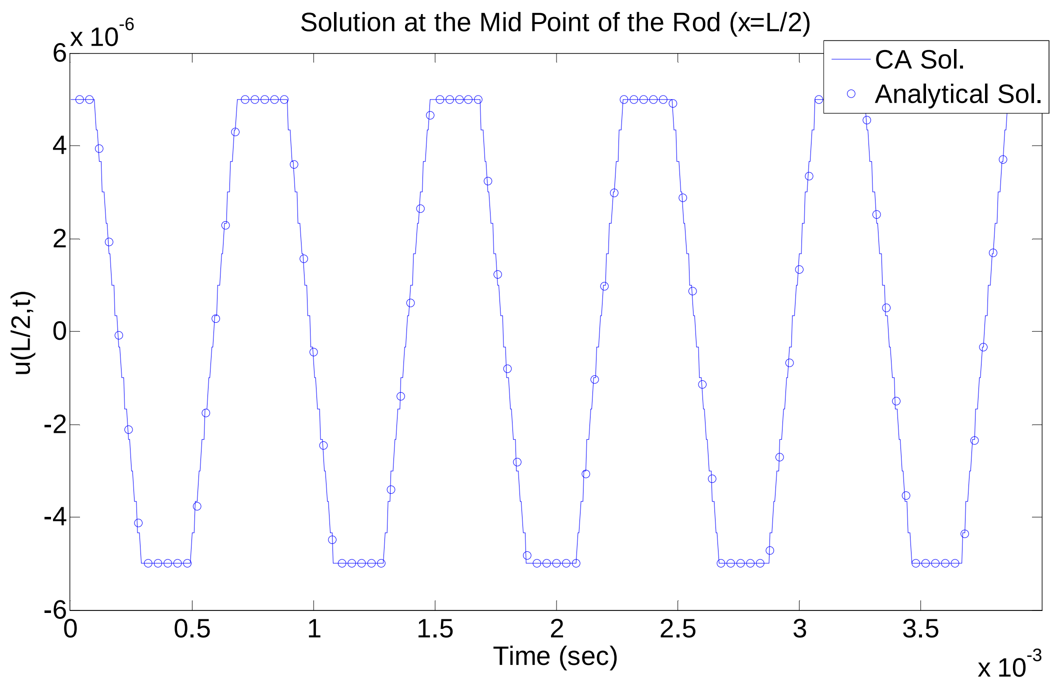


Figure 12. CA and analytical solution of rod problem at x=L/2.

The longitudinal vibration of the rod is governed by the one dimensional wave equation given in Eq.22. The CA approach and local time evolution rules we defined in Chapter II successfully modeled the behavior of the macroscopic behavior of a physical system governed by Eq.22. In the next section, we will try to model another system governed by the one dimensional wave equation.

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{22}$$

C. VIBRATING STRINGS

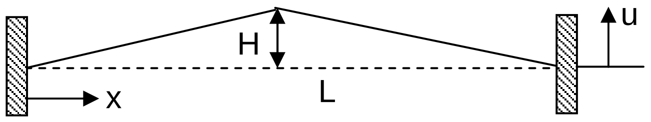
1. String Plucked at the Midpoint

To further examine the ability of the CA model to accurately model a more complex physical system, our next challenge is to model a vibrating string where both ends are fixed. We may think of stringed musical instruments as examples. The string is initially plucked at the middle and released at $t = 0$. The physical system and the corresponding CA model are shown in Figure 13.

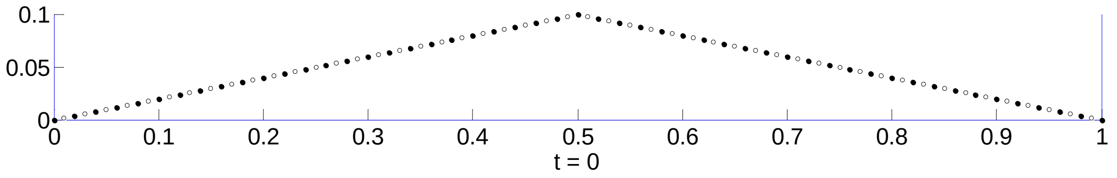
There are some points that we must mention. First is the boundary conditions. In the previous example only left end of the rod was fixed, but in the string problem both ends are fixed, so the first and last particles in the CA model must be defined as constrained particles. Second is the *Time Scaling Factor (TSF)*, as defined in Eq.19 TSF includes the speed of sound (***c_r***) term in the denominator. In the string problem the speed of the sound and the *TSF* is defined as

$$c_r = \sqrt{\frac{T_o}{\rho}} \rightarrow TSF = \frac{L}{(N-1)c_r} = \frac{L}{(N-1)\sqrt{\frac{T_o}{\rho}}} \tag{23}$$

where ***T_o*** represents the tension of the string, which is assumed to be constant along the string and ***ρ*** represents the mass of the string per unit length.



(a) String plucked at midpoint



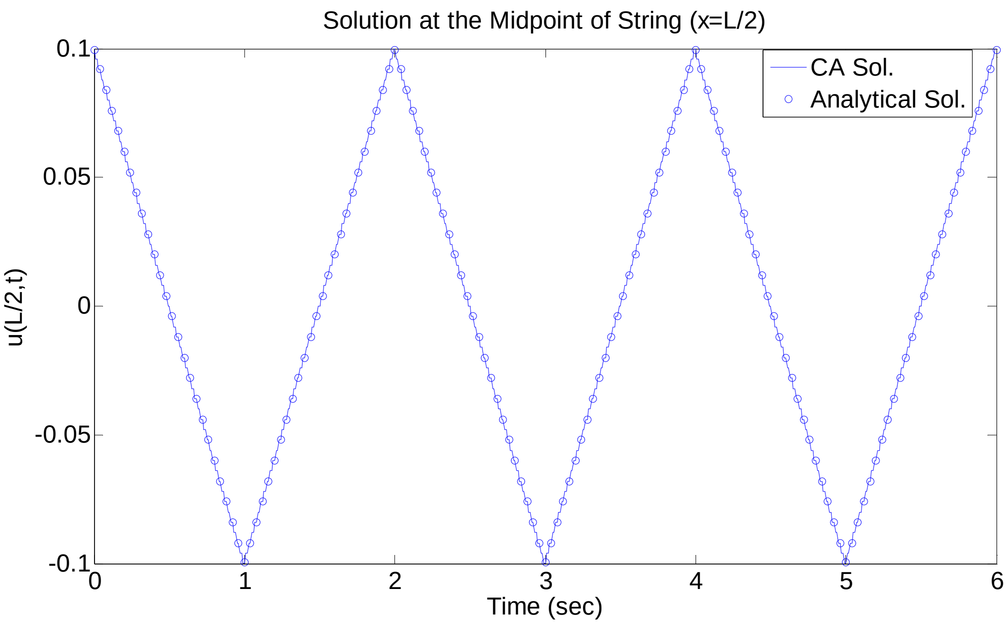
(b) CA model of the string problem

Figure 13. String problem and corresponding CA model.

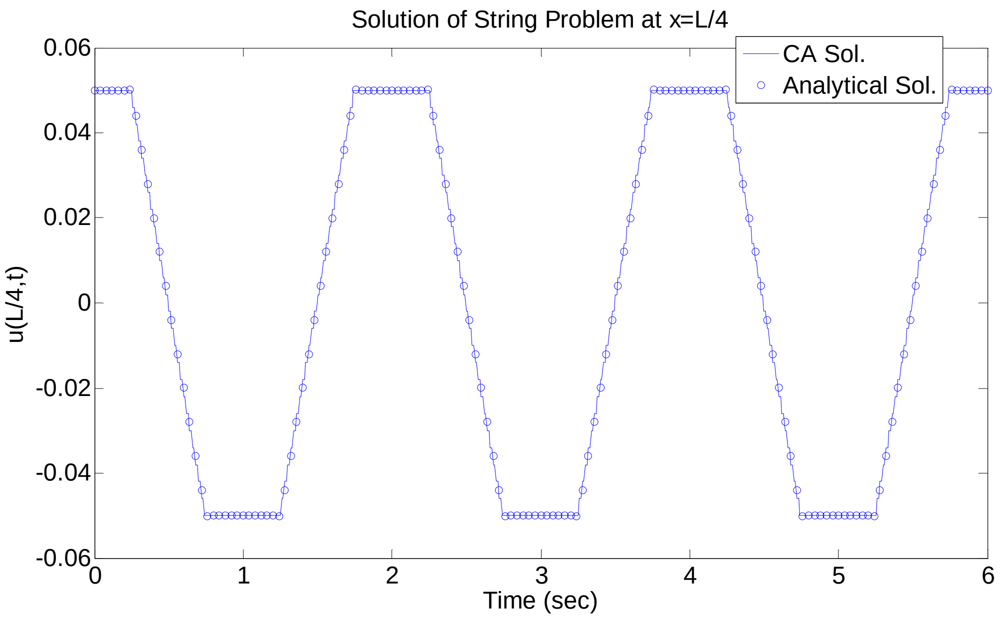
The string is initially plucked at the midpoint and released with zero initial velocity. The initial displacement at the midpoint is ***H = 0.1 m***, the length of the string is ***L = 1 m*** and speed of sound in the string is ***c_r = 1 m/s*** (arbitrary choice). The total number of particles is ***N = 101*** (51 black and 50 white). The analytical solution [13] is given as

$$u(x,t) = \frac{8H}{\pi^2}\sum_{n=0}^{\infty}\frac{(-1)^n}{(2n+1)^2}\sin\frac{(2n+1)\pi x}{L}\cos\frac{(2n+1)\pi ct}{L} \tag{24}$$

The comparison of the analytical and CA solutions is shown in Figure 14 at $x = L / 2$ and $x = L / 4$. The CA solution again matches to analytical solution for the string problem. This shows that the proposed CA rule and the CA modeling approach can be used to model physical systems governed by the one dimensional wave equation.



(a) x = L/2



(b) x = L/4

Figure 14. CA and analytical solution of the string problem.

Up to this point we have considered only displacement boundary conditions. However, we need to find a method to implement force boundary conditions as well as displacement boundary conditions. Our next example is the vibration of a string with a force acting on it.

2. String with a Force Acting on the Middle

As previously discussed, we must be able to implement force boundary conditions on a CA model. Basically we will use Newton's 2nd law to apply the forces. The first step is to discretize the total mass of the string by modeling it as lumped masses connected with springs. The white and black particles in the CA model represent these lumped masses. According to the CA definition, the particles at the two ends of the string have half the mass of internal particles. So

$$\begin{aligned} m_i &= \frac{M}{(N-1)} \quad , \text{ for internal particles} \\ m_i &= \frac{M}{2(N-1)} \quad , \text{ for end particles} \end{aligned} \tag{25}$$

where $\boldsymbol{m_i}$ represents mass of each particle, $\boldsymbol{M}$ represents the total mass of the string and $\boldsymbol{N}$ is the number of particles. In order to apply external forces to the CA model, the following algorithm is developed

$$\begin{aligned} a_i(t+1) &= \frac{F_i(t+1)-F_i(t)}{m_i} \\ v_i(t+1) &= v_i(t) + a_i(t)\Delta t \\ p_i(t+1) &= p_i(t) + v_i(t+1)\Delta t \end{aligned} \tag{26}$$

where $\boldsymbol{F_i}$ is the force applied to the $\boldsymbol{i^{th}}$ particle, $\boldsymbol{m_i}$ is the lumped mass, $\boldsymbol{a_i}$ is the acceleration, $\boldsymbol{v_i}$ is the velocity and $\boldsymbol{p_i}$ is the position of the particle. Once the displacements of the CA particles which the forces are applied are computed by using Eq.26, the local rules given in Eq.2-3-4 are applied to all particles. Once these steps are complete, the above process is repeated.

In our example, physical properties of the string are given as $\boldsymbol{M = 10^{-2}\ kg}$, $\boldsymbol{L = 0.5\ m}$, $\boldsymbol{c_r = 0.5\ m/s}$ (arbitrary) and a sinusoidal force is applied at the midpoint of the string (Figure 15). The forcing function is given as $\boldsymbol{F(t)=0.05*\sin(10t)}$.

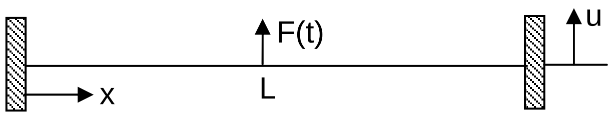
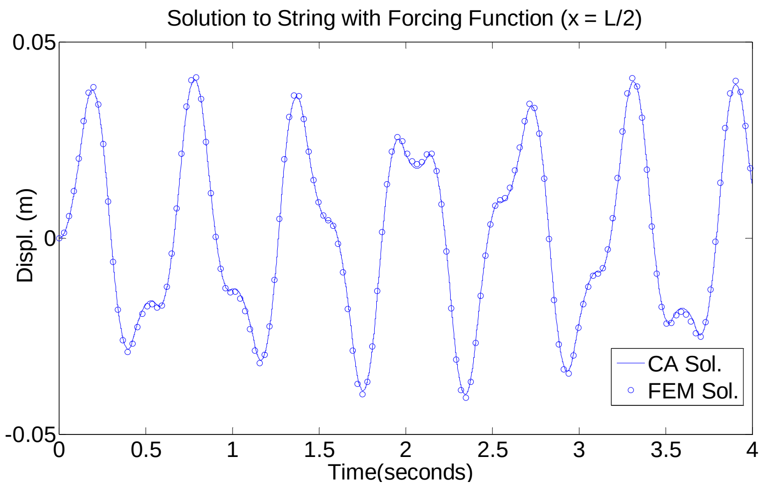
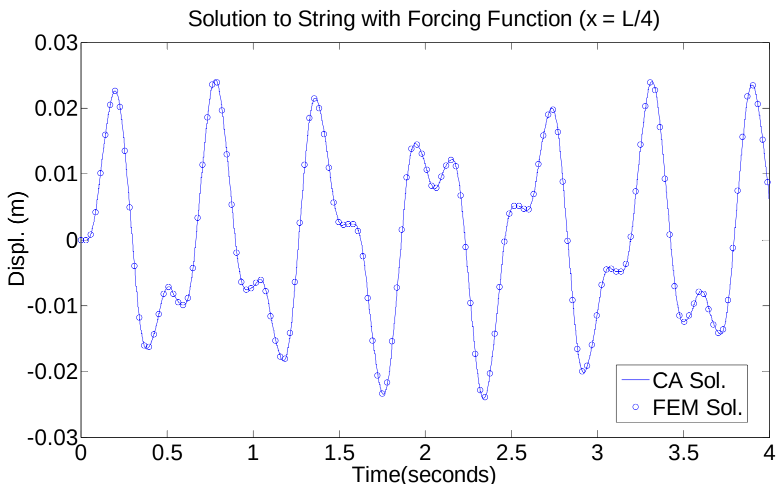


Figure 15. String with a forcing function.

The solution and the comparison at x = L/2 and x = L/4 is shown in Figure 16.



(a) x = L/2



(b) x = L/4

Figure 16. Solution of string with forcing function problem.

The CA solution again matches with the analytical solution, however, there is an interesting point when the mass of the string is involved in the problem. The *TSF* is no longer valid (in the form we defined earlier) in the problems where external forces are involved. The new form of the *TSF* turns out to be

$$TSF = \frac{L}{(N-1)\sqrt{\frac{T_o}{\rho}}}\sqrt{\frac{M}{L}} = \frac{L}{(N-1)c_r}\sqrt{\frac{M}{L}} = \frac{\sqrt{LM}}{(N-1)c_r} \tag{27}$$

This formula or the additional multiplier (which is the square root of the linear mass density of the string) results from experimentation / trial and error and does not have a rigorous mathematical basis, however, it successfully models physical behavior in string problems where external forces are applied. Due to the lack of related research in the literature, it is noted that this is an area that should be investigated further.

To further demonstrate that *TSF* works for string problems with forcing functions, we present another example. In this case, there are two forces acting on the string, described by the following forcing function: $F_1(t)=0.05*\sin(10t)$ and $F_2(t)=0.03*\sin(12t)$, acting at $x=L/4$ and $x=3L/4$, respectively. The physical quantities of the related system are given as, $L = 2.5m$, $M = 0.01\ kg$ and $c_r = 1.5\ m/s$ (arbitrary). Figure 17 shows the comparison of the solutions of CA and Finite Element Method (FEM) at the midpoint of the string.

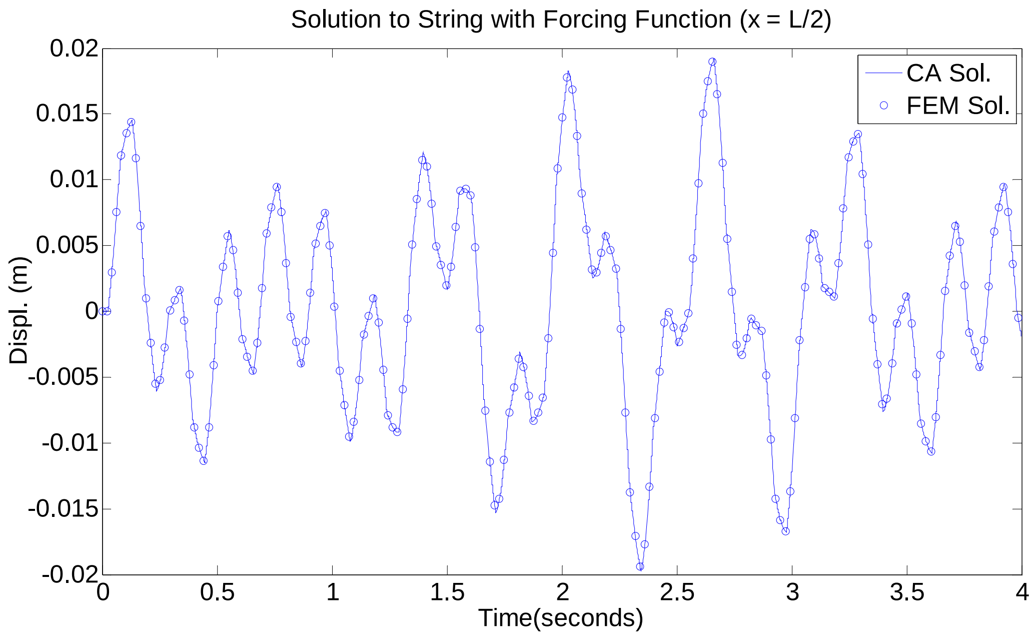


Figure 17. Solution of string with two forcing functions.

The CA solution matches the FEM solution, and we showed that the TSF works as it is given in Eq.27. In the next section, we will attempt to develop a method for making the CA and FEM models work together.

D. COUPLING CA AND FEM

Our next challenge is to model a physical system by using both CA and FEM approaches and make them work coupled. For instance, the left half of the spring may be modeled with the CA approach and the right half of the string with FEM. In doing so, there are several issues that must be resolved. The most important one is to determine how to pass data between the FEM and CA models. Since we are modeling the string with both approaches, there should be a shared node (Figure 18). Secondly we must determine what data (force, velocity, position etc.) should be passed between CA and FEM.

In the model, the last particle of the CA side is the first node of the FEM side. To calculate the next position of this node we define its neighbors. It is obvious that the left neighbor is the white particle to the left of this node, but it does not have a right neighbor. Since in the physical system this node is not at the boundary we cannot use Eq.4 to calculate the next position of this particle. Therefore, we must use the second node (since the first node is shared) of the

FEM part as its right neighbor. This places an extra requirement that the horizontal node spacing in the FEM should be same as the CA part (in the initial configuration) so that the spring vector is still valid for this shared node and its right neighbor (2nd FEM node).

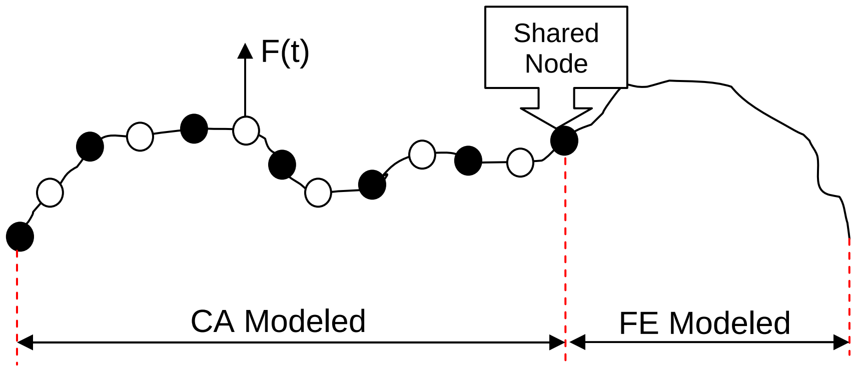


Figure 18. CA and FEM coupled string.

The procedure for computing the coupled CA and FEM calculations is as follows

1. Define the physical problem; divide the domain into the CA and FEM parts. Define the CA and FEM nodes, and make the node at the interface a shared node.

2. Apply any required force by using Eq.26 and Eq.27.

3. Apply the CA rule to the CA modeled part of the string. For calculating the next position of the last particle (shared node) we need the displacement of two neighboring particles. Since this is the last node there is no particle on the positive side. This data is obtained from the second node of the FE model.

$$\begin{aligned} [r_+(t)]_{Last\ node\ of\ CA} &= [r(t-\Delta t)]_{2^{nd}\ node\ of\ FEM} \\ [r(t+\Delta)]_{Last\ node\ CA} &= r_+(t) + r_-(t) - r(t)_{Last\ node\ CA} \end{aligned} \tag{28}$$

4. For $\boldsymbol{t = 0}$ we need the FEM data at $\boldsymbol{t = -\Delta t}$. We will assume that the data at $\boldsymbol{t = -\Delta t}$ is equal to the data at $\boldsymbol{t = 0}$.

5. Pass the displacement data of the last particle of the CA part to the first node of the FEM part, and set the acceleration of this node to zero so that it cannot move during the FEM calculations (This works like a boundary condition).

$$\begin{aligned}[r(t)]_{1^{st}\ node\ of\ FEM} &= [r(t)]_{Last\ node\ of\ CA} \\ [acc(t)]_{1^{st}\ node\ of\ FEM} &= 0\end{aligned} \tag{29}$$

6. Apply FEM calculations as usual.

7. Go back to step 2.

While using this procedure ensure that the time step size for FEM calculations is same as the CA calculations by setting $dt = TSF$, so that FEM and CA calculations are synchronized.

An example of modeling a string system by using both CA and FEM is shown in Figure 19. The physical quantities are given as $L = 1.5\ m$, $M = 0.01\ kg$, $c_r = 1.5\ m/s$ (arbitrary). Total number of nodes N is 101 where 25 of them are modeled as CA nodes (N_{CA}) and 78 of them are modeled as FEM nodes (N_{FEM}) (39 at the right side and 39 at the left side, 2 of them is shared). The force is applied at $x = L/2$ and given as $F(t) = 0.05\ sin(10t)$.

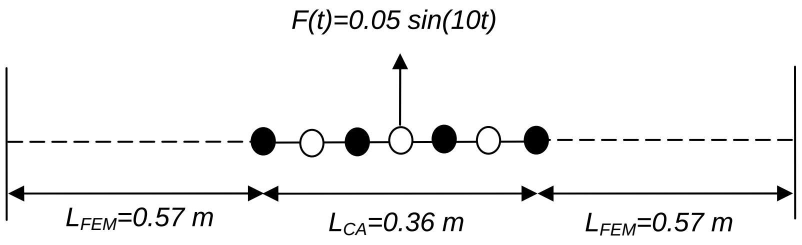
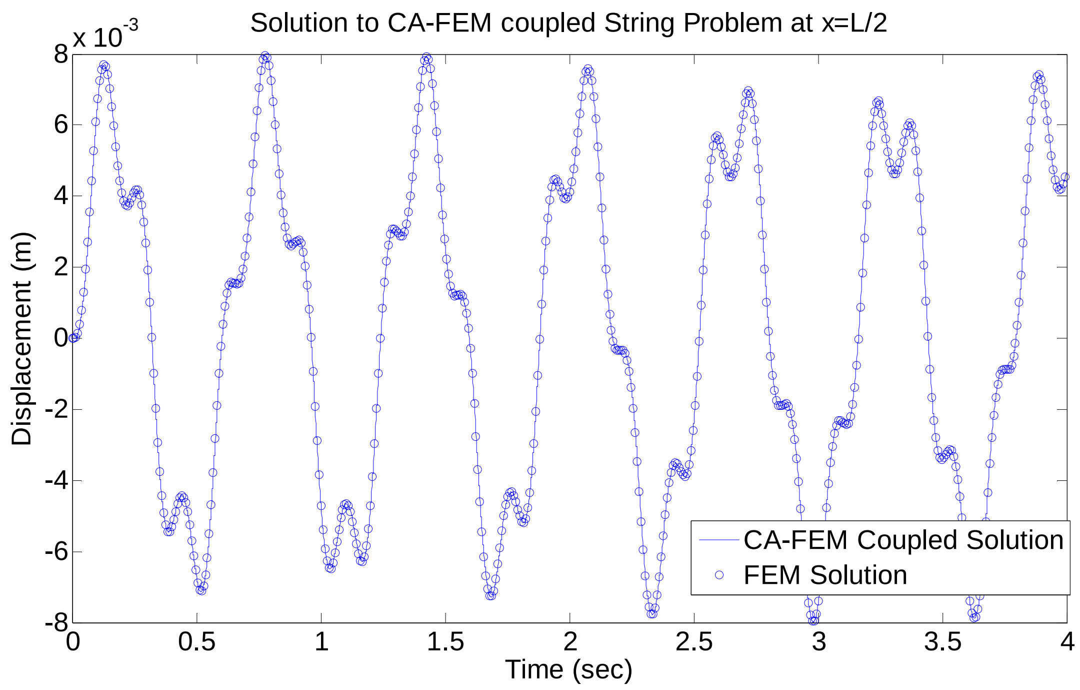
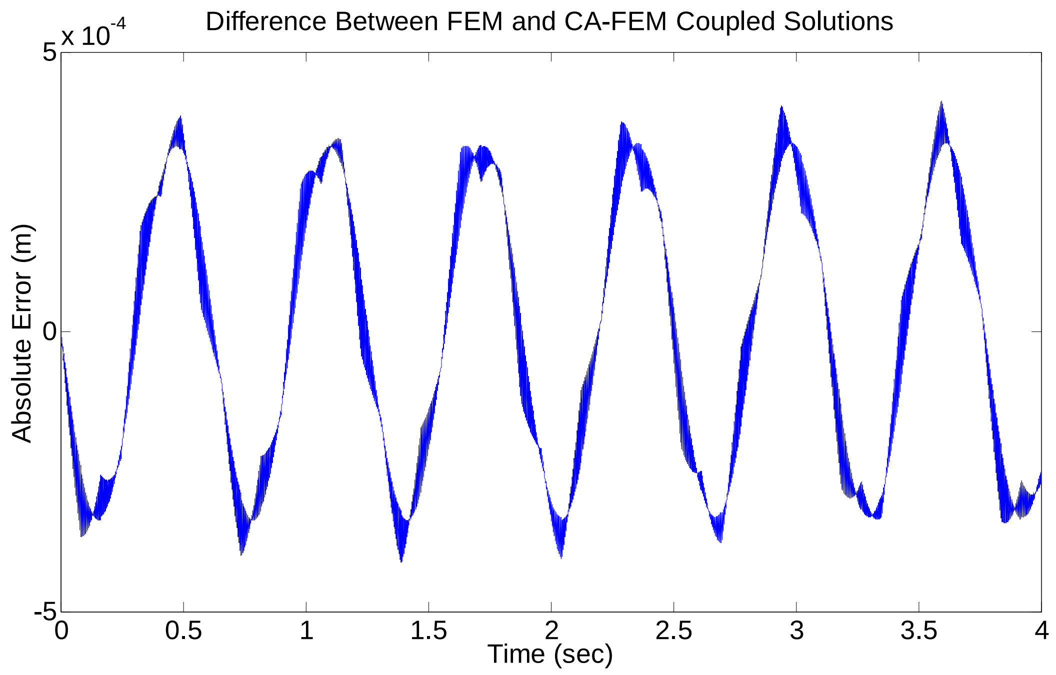


Figure 19. CA and FEM coupled string problem.

After running the CA model, the results are shown in Figure 20. The relative error is about 4%, assuming the FEM solution is correct.



(a) Solution at x = L/2



(b) Absolute errors

Figure 20. Displacement of mid node with CA-FEM coupling and absolute errors

Another possibility is to overlap the CA and FEM portions of the problem at the interface and determine if this decreases the relative errors. Figure 21 shows the calculated absolute errors with and without overlapping which are almost the same. It is seen that overlapping does not provide extra precision to the calculations.

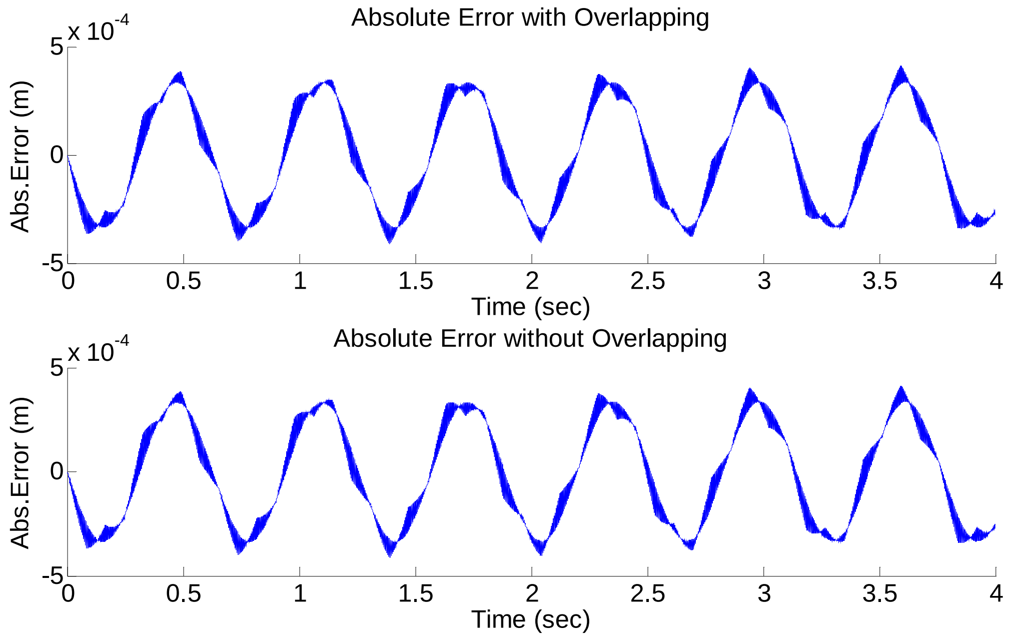


Figure 21. Absolute errors with and without overlapping

In this chapter, we have attempted to develop a methodology for modeling one degree of freedom systems using the CA model described in Chapter II. We have shown that the proposed CA rule can successfully model the systems governed by the one dimensional wave equation. Our first goal was to determine whether a CA model can represent a real physical system at a macroscopic level. After matching the solution at the spatial level, we introduced the *Time Scaling Factor* to enable us to match the temporal part of the real solution with the analytical solution (Eq.19). However, the *TSF* did not work as we defined it when forces are involved. The second definition of *TSF* was derived by a trial and error method (Eq.27). This second formulation does not have a rigorous mathematical basis and therefore requires further investigation. The final section of the chapter was about a CA and FEM coupled model of a string, and we showed that the proposed CA rule and coupling technique can give the same results as a FEM model alone. Noting the successful modeling of one dimensional, one degree of freedom real physical systems with the CA approach, in the next chapter we will attempt to implement this same CA methodology on two dimensional systems. The two dimensional problems may be again one degree of freedom systems but the CA rule is going to be different for two dimensions, namely CA models that each particle has 4 neighbors (except the boundary particles).

THIS PAGE INTENTIONALLY LEFT BLANK

IV. MODELING TWO DIMENSIONAL PROBLEMS

A. THE CA RULE IN TWO DIMENSIONS

In a two dimensional (2D) coordinate system, each particle has four neighboring particles, except the boundary particles. These particles are named east, west, north and south neighbors and constitute a von Neumann neighborhood. The subscripts ‘c’, ‘n’, ‘s’, ‘e’ and ‘w’ represent center, north, south, east and west, respectively. By this convention there are two kinds of springs with different orientations that link particles. One is oriented in the positive x-direction ($\boldsymbol{a}_{we}$), the other is oriented in the positive y-direction ($\boldsymbol{a}_{sn}$), where east and north directions are arbitrarily assumed to be positive (Figure 22).

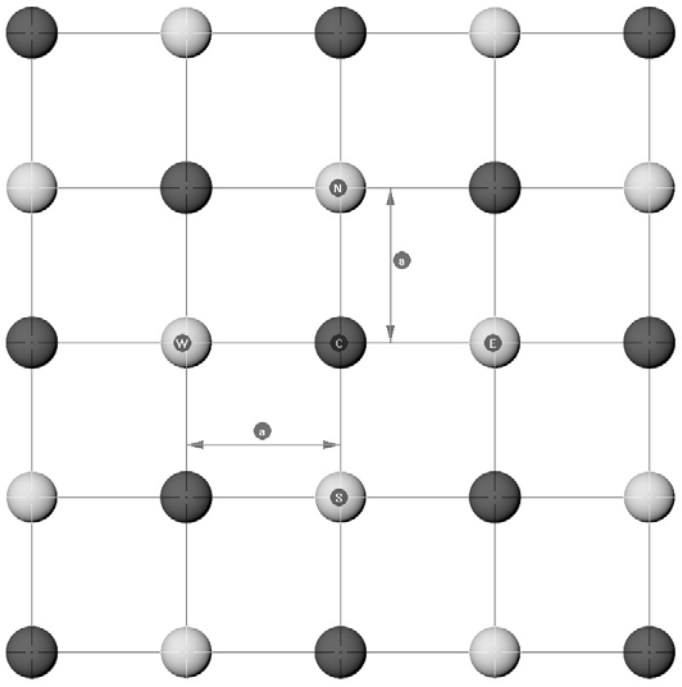


Figure 22. Lattice and particles in a two dimensional CA model

For a particle surrounded by exactly four neighboring particles, the evolution in time, or CA rule, is expressed as

$$r_c(t+\Delta t) = \frac{r_e(t)+r_w(t)+r_n(t)+r_s(t)-2r_c(t)}{2} \tag{30}$$

Equation 30 is an expansion of the local CA rule that we defined for one dimensional grids (Eq. 2). For particles at the east side boundary with three neighboring particles, the evolution in time, or CA rule, is expressed as

$$r_c(t+\Delta t) = \frac{2(r_w(t)+a_{we}) + r_n(t) + r_s(t) - 2r_c(t)}{2} \tag{31}$$

Similarly, the rules for west, north and south boundary particles are respectively

$$r_c(t+\Delta t) = \frac{2(r_e(t)-a_{we}) + r_n(t) + r_s(t) - 2r_c(t)}{2} \tag{32}$$

$$r_c(t+\Delta t) = \frac{2(r_s(t)+a_{sn}) + r_e(t) + r_w(t) - 2r_c(t)}{2} \tag{33}$$

$$r_c(t+\Delta t) = \frac{2(r_n(t)-a_{sn}) + r_e(t) + r_w(t) - 2r_c(t)}{2} \tag{34}$$

Finally, there is a different rule for a corner particle. For example, the rule for the northeast boundary of the lattice, the rule is

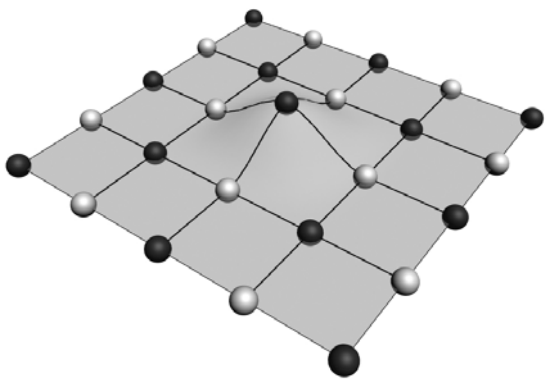
$$r_c(t+\Delta t) = \frac{2(r_s(t)+a_{sn}) + 2(r_w(t)+a_{we}) - 2r_c(t)}{2} \tag{35}$$

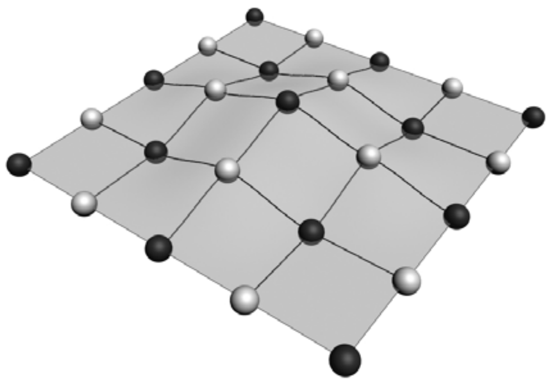
Similar rules can be developed for other corners of the grid. In the above equations, $\boldsymbol{a_i}$ ($\boldsymbol{i=we}$ or $\boldsymbol{sn}$) is the spring vector constant that gives the orientation and equilibrium length of the springs at each direction. $\boldsymbol{r_i}$ vector is the position vector of each particle with respect to a given origin point. The time evolution of a 5 by 5 CA grid, with an arbitrary initial displacement at the center node, and with all edges constrained, is demonstrated in Figure 23.

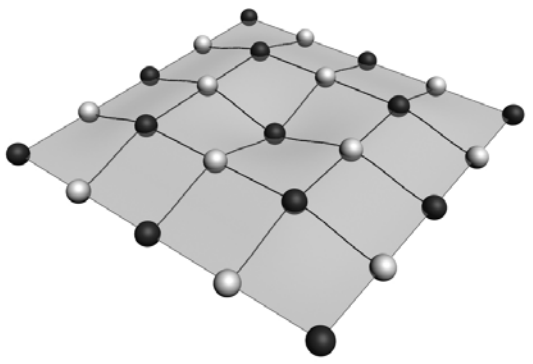
The equations given above define the spatial time evolution of the particles, but for the temporal part to match the real-time scale, we must define a *Time Scaling Factor* (*TSF*) for 2D domain problems. The TSF for 2D problems is

$$TSF = \sqrt{\left(\frac{L_x}{(N_x - 1)}\right)^2 + \left(\frac{L_y}{(N_y - 1)}\right)^2}\sqrt{\frac{M}{L_x L_y}}\frac{1}{2c} \qquad (36)$$

where $\boldsymbol{L_x}$ and $\boldsymbol{L_y}$ are length of the domain in the x and y directions, $\boldsymbol{N_x}$ and $\boldsymbol{N_y}$ are the number of particles along the x and y directions, $\boldsymbol{c}$ is the speed of sound and $\boldsymbol{M}$ is the total mass.


(a) *t = 0*


(b) *t = 1*


(c) *t = 2*

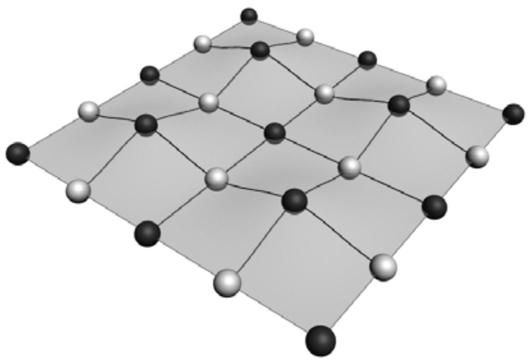

(d) *t = 3*

Figure 23. Time evolution of a 2D CA grid

Having defined the 2D CA equations and discussed the associated 2D methodology, in the next section, we give an example of a 2D membrane problem with an initial displacement at the center.

B. MEMBRANE PROBLEM WITH INITIAL DISPLACEMENT

Figure 24 depicts the physical problem. The membrane is 1.7 m long in both the x and y directions, the center node is initially displaced 0.1 m, the mass of the membrane is 0.2 kg and the number of nodes is 41 in both the x and y directions (total of 41x41 = 1681 nodes).

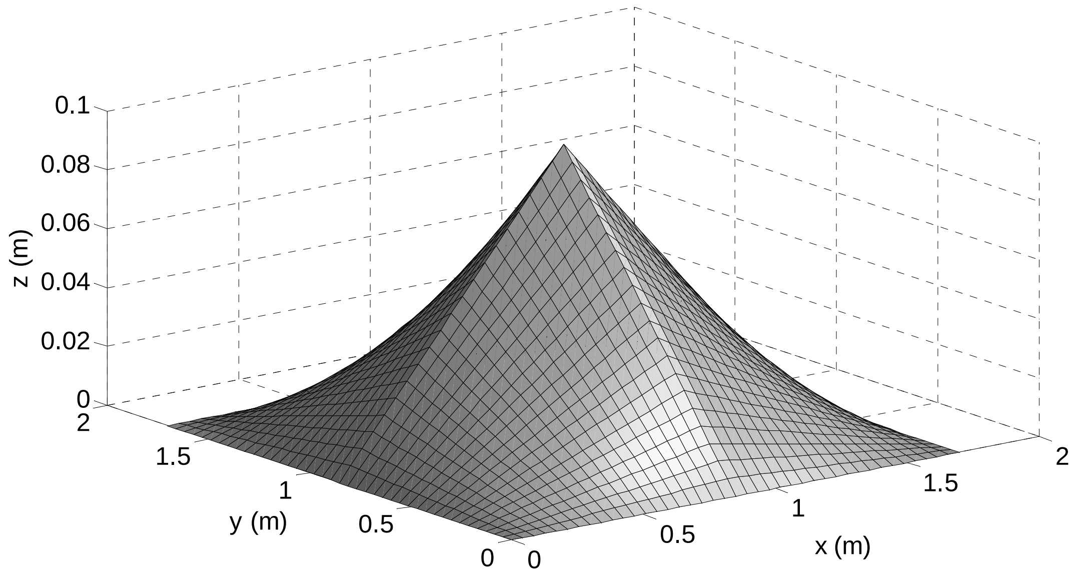


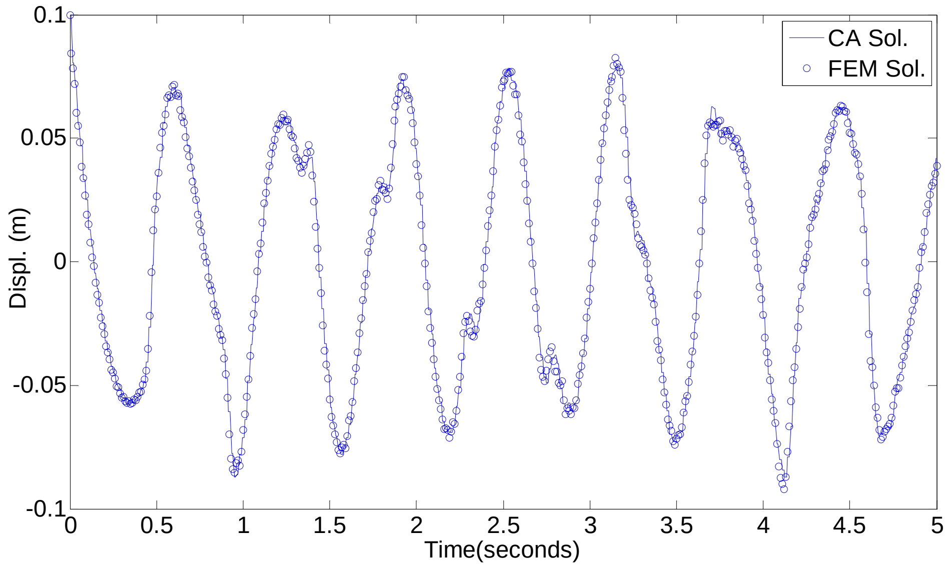
Figure 24. Initial configuration of membrane.

The membrane is clamped on each of its 4 edges, and released from this initial configuration with zero velocity and acceleration. By using Eqs.30-35 and CA time evolution methodology (black and white particles), we can easily model this membrane. Figure 25 shows a comparison of the displacement solutions at the center and at node 345 (arbitrary) for both the CA and FEM methodologies. Again the CA solution agrees with the FEM solution.

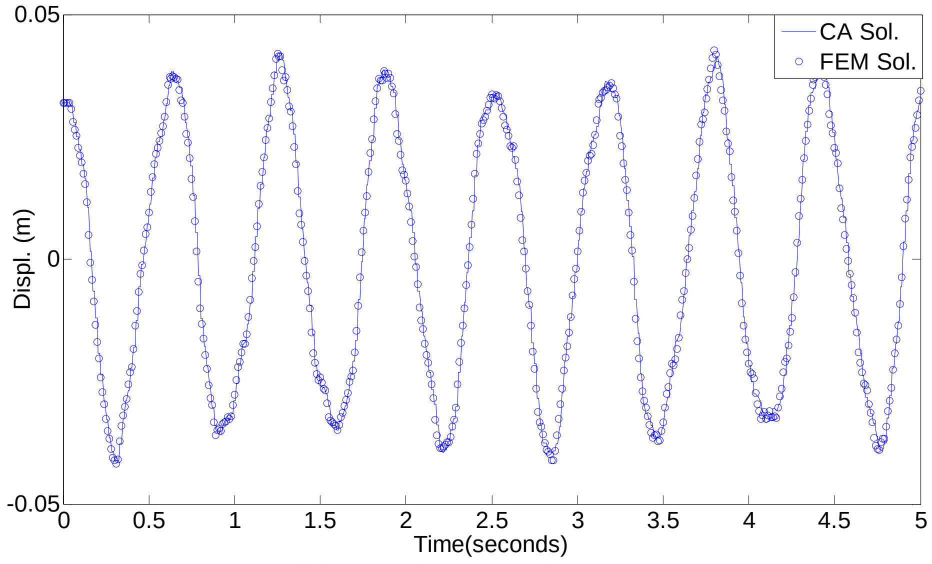
At some point, one may question the use of CA when there is a well-developed method like FEM (besides scientific curiosity). First of all, CA is very intuitive and easy to implement. CA calculations consist of only four basic algebraic operations. It is very memory efficient because there are no large system matrices. Most importantly, CA solutions are computed very quickly compared to FEM, especially when the number of nodes increases. For the previous membrane problem, Table 1 shows the MATLAB calculation times of the CA and FE methods for different numbers of nodes.

Number of Nodes	CA Calculations (sec)	FEM Calculations (sec)
11x11 = 121	0.24	0.38
21x21 = 441	1.7	4.9
31x31 = 961	5.85	56.82
41x41 = 1681	14.3	250.1
51x51 = 2601	26.7	775.5

Table 1. Calculation times for CA and FEM



(a) Solution at the center of the membrane



(b) Solution at node 345

Figure 25. Comparison of CA and FEM solutions for the membrane problem.

An approximate number of required addition, subtraction, multiplication and divisions to calculate each time step in CA and FEM (central difference scheme) is given in Table 2, where s is the system degrees of freedoms which is the degrees of freedoms time the number of nodes.

	CA	**FEM**
Addition	3s	$2s^2+3s$
Subtraction	s	3s
Multiplication	s	$2s^2+8s$
Division	s	-

Table 2. Number of required algebraic calculations

According to Table 2, it can be seen that with an increasing number of nodes that the calculation time for FEM increases quadratically, whereas it increases linearly for CA. Another important factor is that the required number of clock cycles to complete a multiplication or division is approximately 3 to 4 times more than that for addition or subtraction, which is another drawback of FEM when compared to CA since the required number of multiplications is related to the square of the system degrees of freedoms in FEM calculations. When modeling very complex and large systems, it is not uncommon to have thousands of nodes and this makes the CA approach a good competitor of FEM. Since every computer system and software has its own way of dealing with calculations (parallel processing, predictive branching, pipelining, scalar/superscalar processing, separate fetching / executing units, etc.), it is not possible to give a formulation for the time required to compute a given problem solution. Figure 26 shows the calculation time vs. system degrees of freedoms for the previous membrane problem. It is possible to fit a first order polynomial for CA and a second order polynomial for FEM calculation times, and they are plotted on top of respective graphs in Figure 26. These graphs support the point that CA calculation times increase linearly with an increasing number of system DOFs and FEM calculation times increases quadratically with an increasing number of system DOFs.

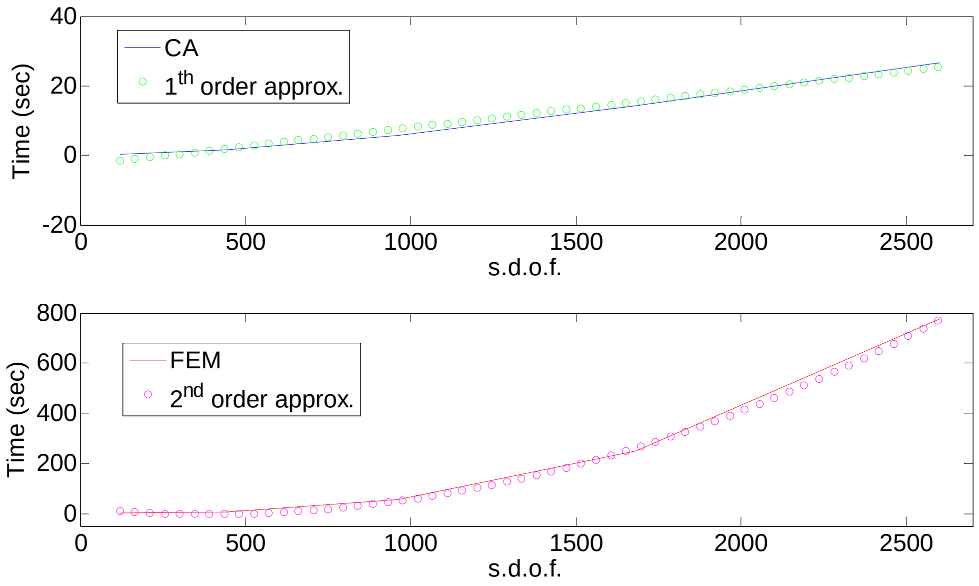


Figure 26. Calculation times versus system degree of freedoms.

C. COUPLING CA AND FEM

CA calculations are faster than FEM calculations, especially when modeling systems with a large number of nodes, as explained in IV.B. However, there are still some issues with CA modeling that are not well defined. Examples include, implementing force boundary conditions and solving static problems. The next step is to investigate whether we can use both methods together in 2D models as we did in 1D problem so that we can exploit the advantages of both CA and FEM. It may be possible to model a large domain in the system with FEM, especially the boundaries and where the forces are applied, and a small domain in the system with CA, where a finer mesh is required and where the CA domain is the actual point of interest. One example may be fracture propagation problems where the point of interest is around the initial crack. This example will be discussed in greater detail in Chapter V.

Consider a simple rectangular membrane problem. The left half of the membrane is modeled with CA and the right half is modeled with FEM. The nodes at the interface are shared nodes and there is no overlapping (Figure 27). A sinusoidal force is acting on the membrane in the middle. As described in the Chapter III, the second column (from the left) of FEM nodes are used as the east neighbors of the last column of CA nodes and the position data are passed from CA to FEM at the interface nodes.

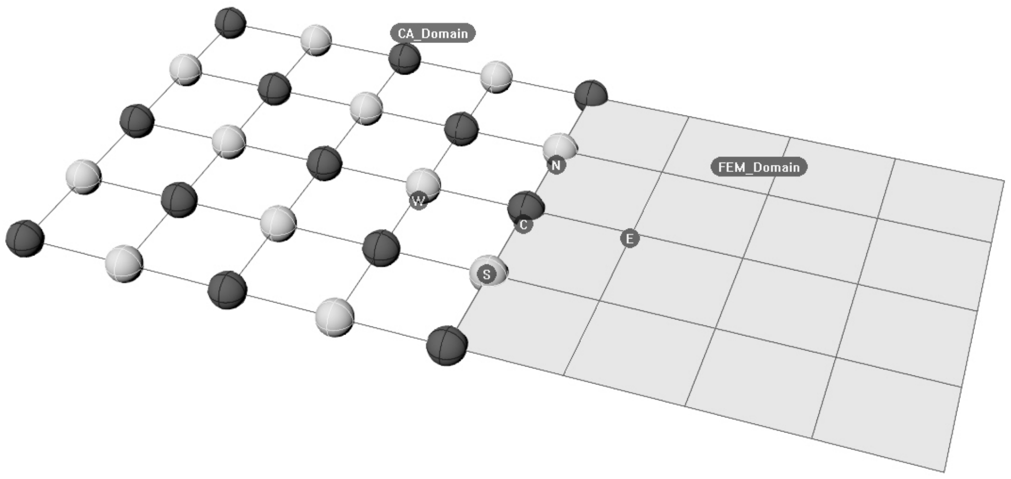


Figure 27. Coupled CA and FEM domains.

The solution at the middle node of the membrane is shown in Figure 28 (top). At this node the maximum relative error is 5.1%. The largest errors appear to be where high frequency fluctuations occur. When a node far from middle is examined (Figure 28, bottom) the solutions seems to match except at the peaks, however, the errors are greater. When the solutions of all the nodes are examined, we can see that at the nodes further from the center the errors are increasing (up to 30%). Two other configurations are also modeled (Figure 29) and gave similar results. In these models, we also observe that the further the nodes are from the CA-FEM interface the greater the errors.

An interesting point is that by changing the TSF, we can reduce or increase these errors. For example, in the previous problem, if instead of using 0.0067 as the TSF we use 0.0065, we can reduce the errors down to 4% at some nodes. This method only assures a good solution at some of the nodes not at all of them. The important result from this is that the TSF, not only depends on the physical properties of the system and the number of nodes, but also depends on the position. Thus, we can say that there is a unique TSF that makes the solution correct for each CA node at a given time. With using the current CA rule (Eqs. 30 through 35) and the coupling methodology introduced above we could not succeed to perfectly couple CA and FE modeling techniques in a membrane problem. At this point, to be able to couple CA and FE techniques, the given CA rule set should be modified or a whole different coupling methodology should be developed.

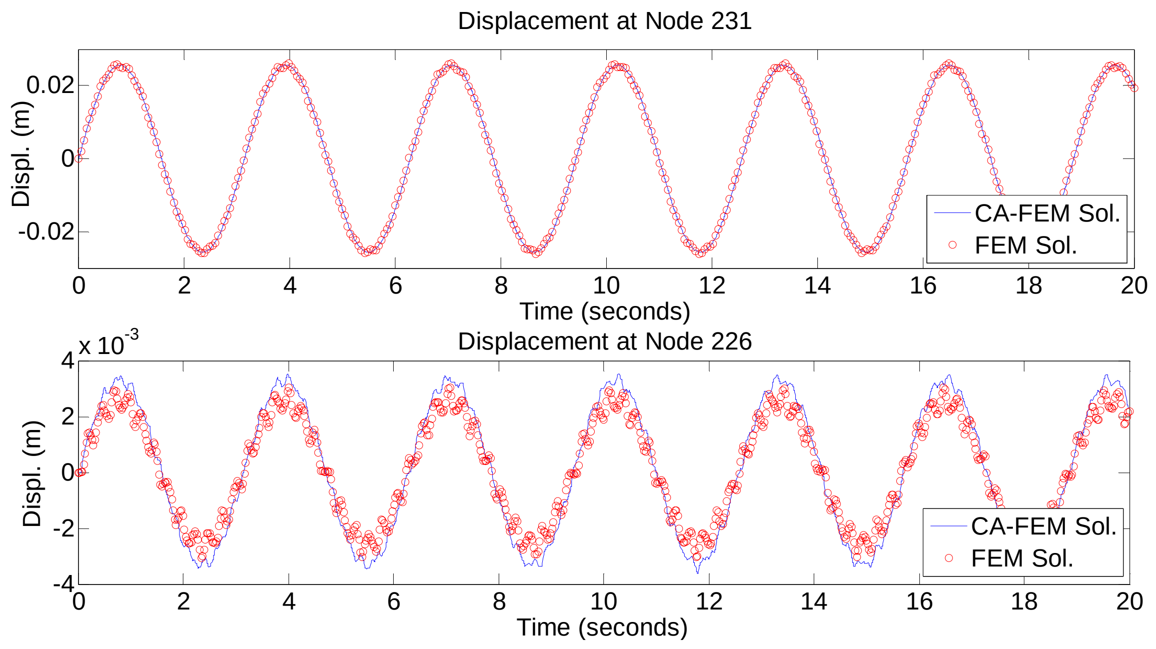


Figure 28. Solution of CA and FEM membrane problem at two different nodes.

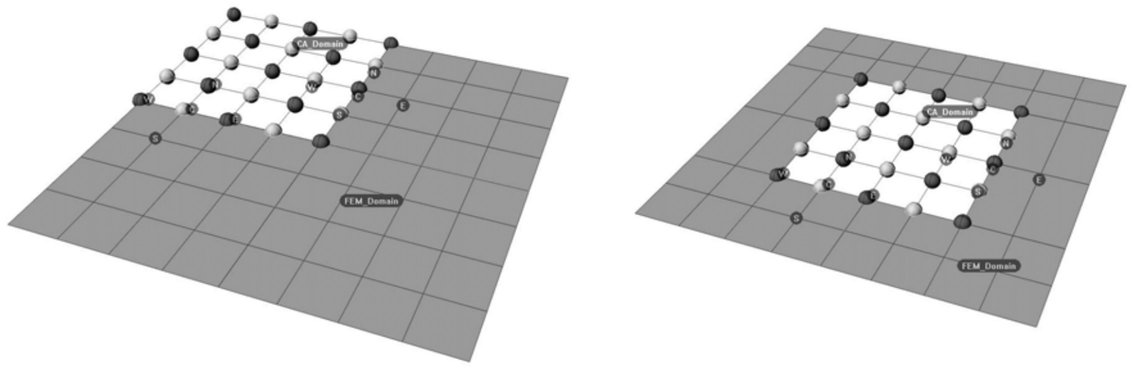


Figure 29. Two other membrane models that was tried to be modeled.

One such alternate methodology is described as follows:

1. Model the system with a crude mesh and solve with FEM. Save the nodal solutions at each time step.

2. Model the part of the problem domain that is the main point of interest by using a fine CA mesh.

3. At each CA calculation step, pass the solution of the FEM nodes that correspond to the boundaries of the portion modeled by CA to boundary CA nodes.

4. Apply the CA rules.

As an example, we modeled an octagonal membrane with a square hole in it. A sinusoidal forcing function acts at the center of the membrane. All eight edges of the membrane are clamped. The membrane is modeled with a crude FEM mesh. The area of interest is the area around the hole, and is modeled with a very fine CA mesh (Figure 30).

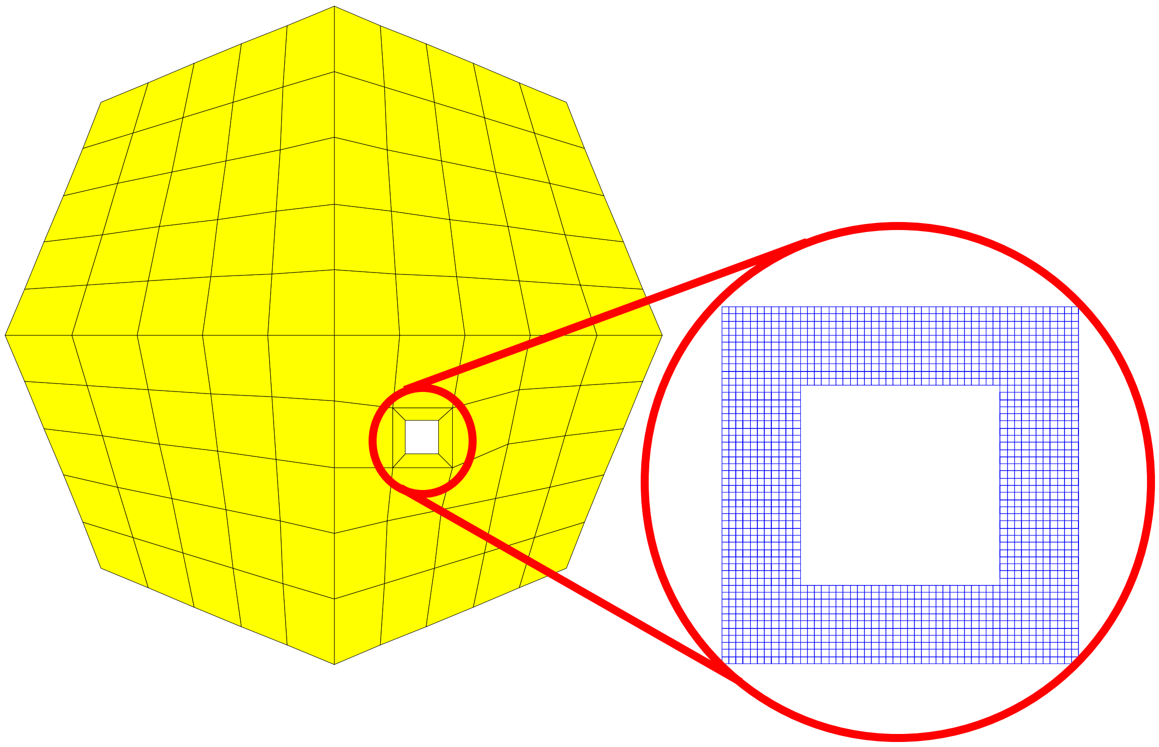


Figure 30. The FEM and CA modeled parts of the octagonal membrane.

First, we solved the whole problem by using only FEM and saved the nodal solutions at each time step. After that, we started CA calculations. At each CA time step, we passed the FEM solutions at the four corners of the interface to the corresponding CA nodes. These solutions were applied as the displacement boundary conditions in the CA model and we applied the CA rules to all other CA nodes. Figure 31 shows the comparison of FEM and CA solutions at the bottom left corner of the hole. The CA-FEM coupled solution is in agreement with the FEM only solution.

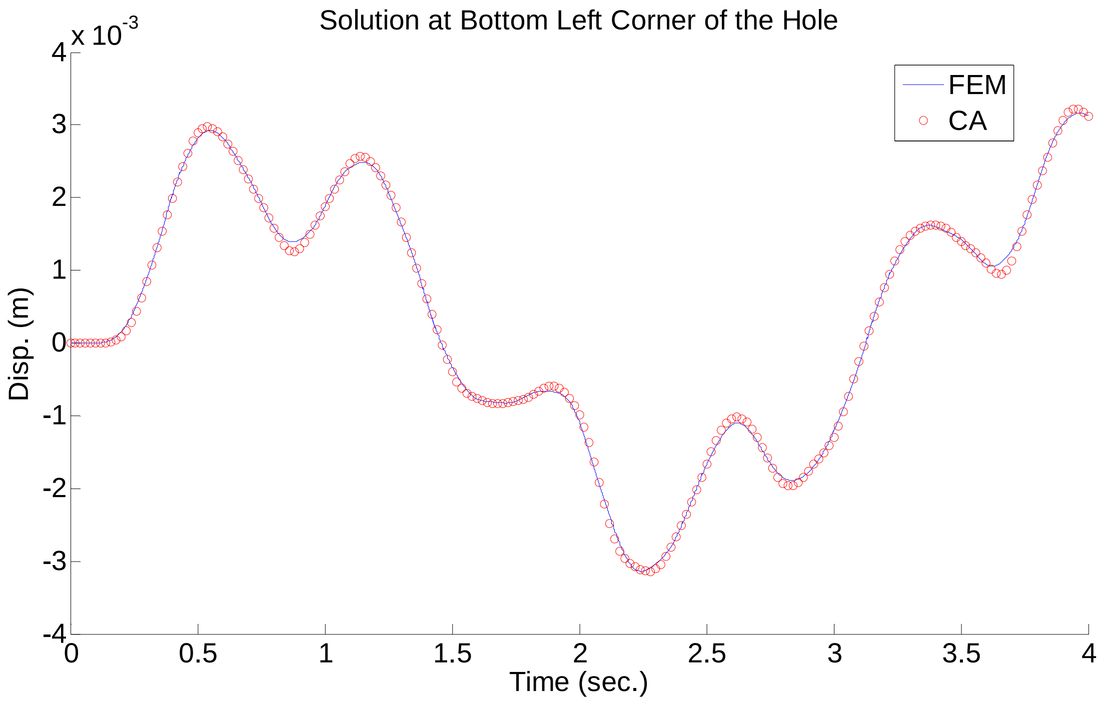


Figure 31. The FEM and CA-FEM coupled solutions at the bottom left corner of the hole.

With this modeling approach, the FEM and CA techniques were successfully coupled in order to solve a two dimensional problem. As a comparison of speed, we also modeled the whole membrane with a very fine mesh and solved with FEM. The fine-meshed FEM model consisted of 2601 nodes and 2500 elements, and the CA-FEM coupled model also consisted of 2601 nodes. The fine-meshed FEM model took 1104 seconds to solve the problem, whereas the CA-FEM coupled solution took 80 seconds. This illustrates that the CA-FEM coupled model approach worked approximately 14 times faster than the FEM approach for this problem, which supports the discussion in Section IV.B.

THIS PAGE INTENTIONALLY LEFT BLANK

V. APPLICATION TO FRACTURE MECHANICS PROBLEMS

A. FRACTURE MECHANICS AT MICRO LEVEL

Simple fracture is defined in [15] as;

> The separation of a body into two or more pieces in response to an imposed stress that is static (i.e. constant or slowly changing with time) and at temperatures that are low relative to the melting temperature of the material. The applied stress may be tensile, compressive, shear or torsional.

For engineering materials there are two types of fractures: ductile and brittle, which are based on the material's ability to undergo plastic deformation. Ductile materials tend to show plastic deformation before fracture, whereas brittle materials generally show little or no plastic deformation before fracture. Ductile materials absorb high energy due to the plastic deformation, while brittle materials absorb very low energy because of the little or no plastic deformation.

Some common equations used in engineering calculations for materials under uniaxial stress are given in Equations 37 through 39.

$$\in = \frac{L_d - L_o}{L_o} = \frac{\Delta L}{L_o} \qquad \begin{array}{l} \mathrm{L_d}: \text{ Deformed length} \\ \mathrm{L_o}: \text{ Original length} \\ \in \ : \text{Axial strain} \end{array} \tag{37}$$

$$\sigma = \frac{F}{A} \qquad \begin{array}{l} \mathrm{F}: \text{Applied force} \\ \mathrm{A}: \text{Cross sectional area} \\ \sigma: \text{Stress} \end{array} \tag{38}$$

$$\sigma = E \in \tag{39}$$

The micromechanics of fracture are considered on the scale of atomic spacing up to grain size. At this level, two types of fractures, cleavage and intercrystalline are relevant. Cleavage fractures proceed along the characteristic planes of the lattice structure so that its orientation changes at the grain boundaries in a polycrystalline material [16]. Cleavage fracture is described as transcrystalline, meaning that it is formed by a separation within the individual grains (Figure 32).

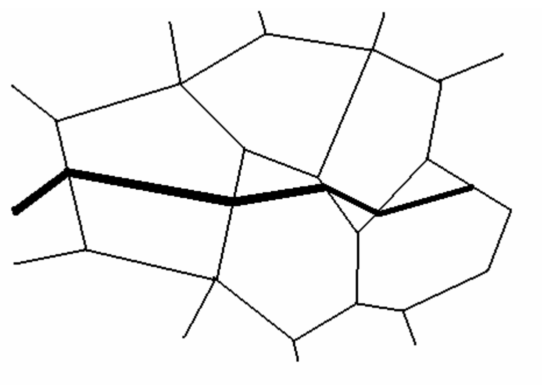


Figure 32. Transcrystalline fracture propagation.

Intercrystalline fractures are mostly due to the anomalies at grain boundaries, such as weakened bonds between grains caused by precipitation of oxides, carbides or sulfides, etc [16], and then the crack may follow the grain boundaries as the path of least resistance. In addition to these, impurities in solids, such as vacancies, substitutional and interstitial impurity atoms, dislocations, external surfaces, grain boundaries, twin boundaries, phase boundaries and stacking faults highly affect the physical properties of the material and hence play an important factor on fracture [15].

At the micro level, the critical stress required to break links between two neighboring planes of atoms and cause fracture is given in [16] as

$$\sigma_c \approx \frac{E}{10} \tag{40}$$

under special circumstances, namely when the material is considered to be made of extremely thin fibers or whiskers, such that the cross sections are nearly homogeneously strained. However, again in [16], it is noted that fracture stress may be smaller by one to three decades. Given this, by using Eq. 39 and assuming elasticity, the critical strain at fracture should be on the order of magnitude of ***0.1 (10%)***.

B. FRACTURE MODELING WITH CA

In this section we attempt to model crack propagation on a solid body. In CA modeling consider the CA particles as the atoms that form the solid. The CA rule set introduced in Chapter IV (Eqs. 30 through 35) has some shortcomings when modeling solid bodies as opposed to wave equations. The first one is the effect of the Poisson ratio and the other is damping. Without damping, particles vibrates infinitely and never come to rest or to a steady state condition. The damping can be included if the position of a particle in the next time step can be written as a correction to the current position of the particle. At this point, a more general CA rule, defined in [3], can be used. The rule given in Eq. 41 and 42 is just a more general formulation of Eq. 30 that can be applied to both center and boundary nodes.

$$\vec{r}_{cm} = \frac{[(\vec{r}_w + \vec{h})n_w + (\vec{r}_e - \vec{h})n_e + (\vec{r}_s + \vec{u})n_s + (\vec{r}_n - \vec{u})n_n]}{(n_w + n_e + n_s + n_n)} \tag{41}$$

$$\vec{r}_c(t+1) = \vec{r}_c(t) + 2[\vec{r}_{cm} - \vec{r}_c(t)] \tag{42}$$

where $\boldsymbol{r_c}$ is the position vector of center node, $\boldsymbol{r_{e,w,n,s}}$ are the position vectors of neighboring nodes, $\boldsymbol{h}$ and $\boldsymbol{u}$ are spring vectors in the x and y directions, respectively, $\boldsymbol{n_{e,w,n,s}}$ are Boolean values indicating the presence of a neighbor along the east, west, north and south directions (Figure 33). These Boolean variables are used to model boundary particles with less than four neighbors and broken links, and take the value of 1 for the presence of neighbors or 0 for the absence of neighbors or presence of broken links. $\boldsymbol{h}$ and $\boldsymbol{u}$ represents the equilibrium length of virtual springs that link the particles in the CA model. To include the damping, leaving Eq. 41 unchanged, we modify the second part of Eq. 42 by applying a damping coefficient $\boldsymbol{\gamma}$, which gives us:

$$\vec{r}_c(t+1) = \vec{r}_c(t) + 2\gamma[\vec{r}_{cm} - \vec{r}_c(t)] \tag{43}$$

The damping coefficient γ can take values between 0.5 and 1, 0.5 being the most damped condition which brings the system to steady state fastest, and 1 being the undamped condition which results in the particles vibrating infinitely.

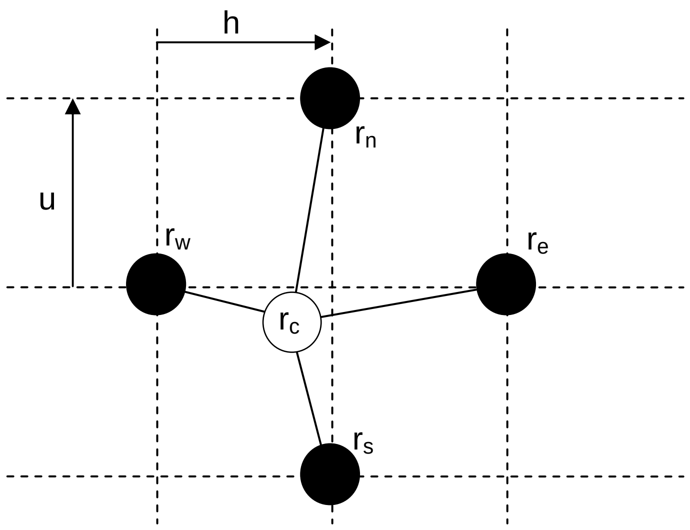


Figure 33. Illustration of CA lattice and particles.

Since we consider the CA particles as the atoms that form the solid body, a fracture can be modeled by breaking the bonds / links of the particles that exceed a given strain or local deformation. After these bonds are broken, these particles act like free surfaces [3]. Recall that our purpose in this section is to determine whether this CA rule can model crack propagation behaviorwise. Note, we are not trying to match some experimental or analytical data.

In this analysis, we model crack propagation on a solid body under uniaxial loading with an initial crack in the bottom middle (some initially broken links). Figure 34 shows the initial configuration of the system. The system is deformed until the first crack starts propagating and is held thereafter. The local deformation criterion we use is the critical strain. When a strain between two neighboring particles exceeds critical strain, which is estimated as 0.1, the link is artificially broken and calculations continue from there.

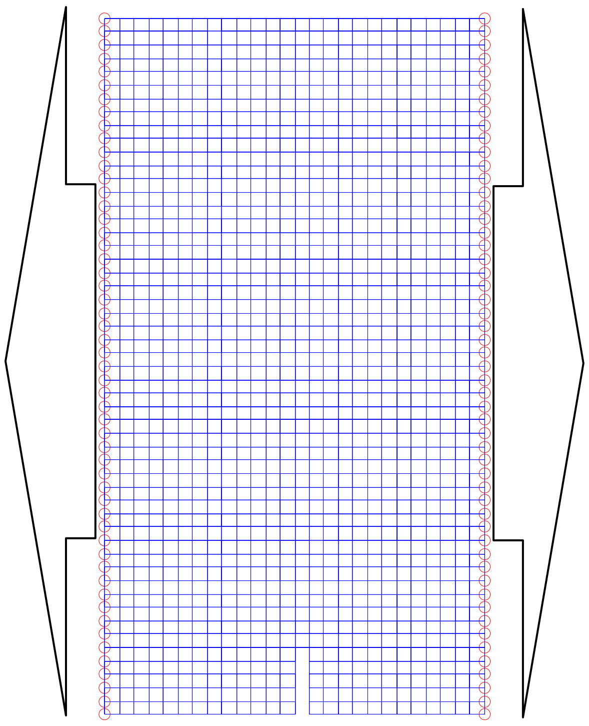
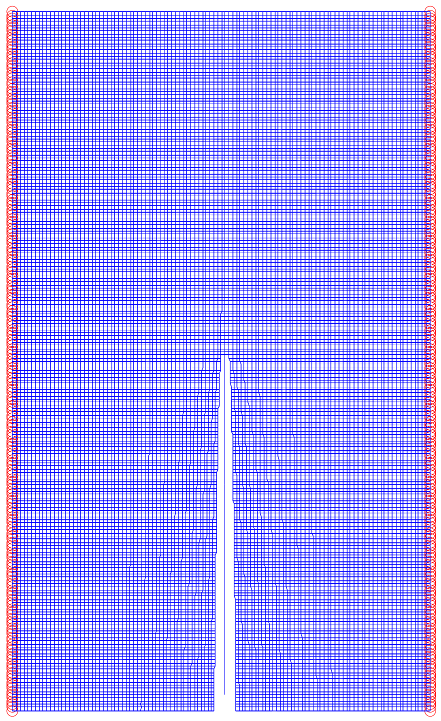
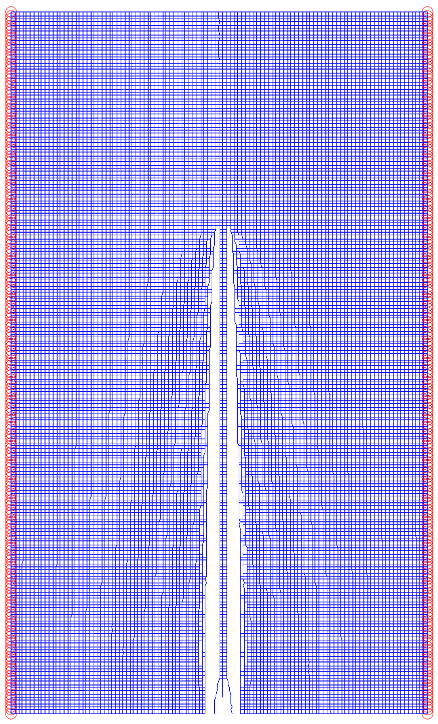


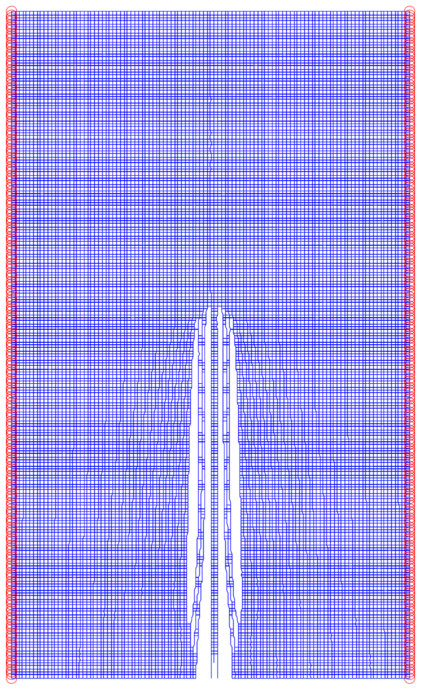
Figure 34. Initial configuration of crack.

According to our definition of the damping factor, a bigger value of γ should represent a more brittle material with more crack branching, and a smaller value of γ should represent a more ductile material with less crack branching. In our strain calculations, we used the engineering strain formula given in Eq. 37. For this analysis, different values for γ were run for the CA model. Figures 35 (a), (b), (c) and (d) show crack propagation patterns when γ is 0.7, 0.85, 0.9 and 0.95, respectively. These figures demonstrate that the bigger the damping coefficient the more brittle the material behavior.

As discussed earlier, in order to model real crack propagation at the micro level, impurities, dislocations and crystal structures should be taken into account. These factors are not modeled with the current 2 dimensional square CA lattice and current rule set. This square lattice configuration is closest to model a Face Centered Cubic (FCC) lattice structure. This model can be improved by using a hexagonal lattice [3] to be able to better model a Hexagonal Closed Pack lattice systems. The success in these experiments is that by using a local rule and local deformations, the behavior of crack propagation at a macro level is captured.


(a) γ = 0.7


(b) γ = 0.85


(c) γ = 0.9

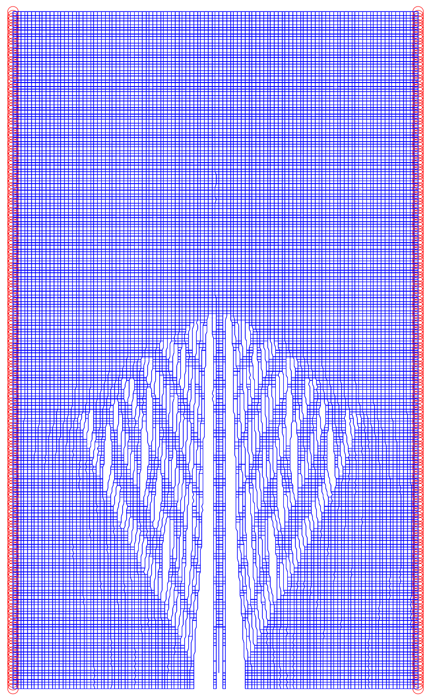

(d) γ = 0.95

Figure 35. Crack propagation with different damping coefficients.

The real power of CA modeling comes from the simplicity and speed of its calculations. In Chapter IV, while trying to couple FEM and CA, our purpose was to develop a methodology to utilize both the speed of CA and ability of FEM to include forces and force boundary conditions, so that we can model the larger section of the problem domain by FEM and the smaller but more important parts of the problem domain with a very fine CA mesh. The crack propagation problem is one of the problems that is well-suited for application of this approach.

For simplicity, a rectangular solid body is going to be modeled and investigated (Figure 36). The body is considered to be under constant uniaxial tension and there is a small initial crack in the bottom center.

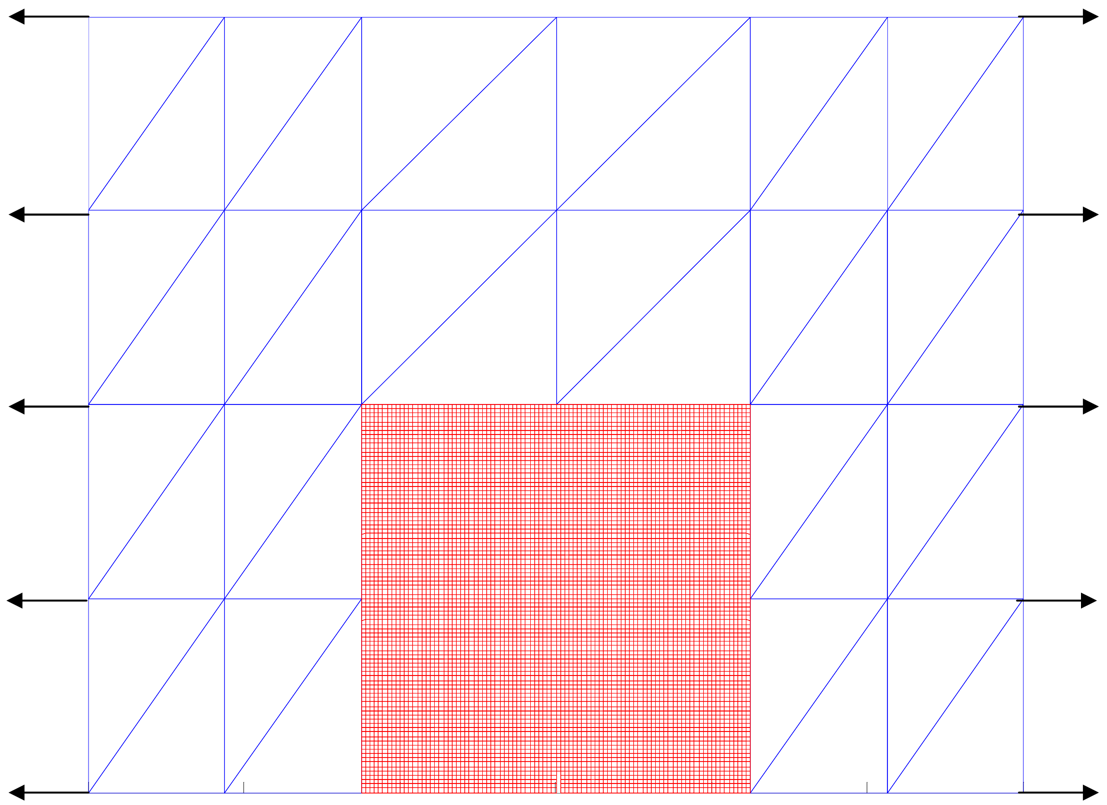


Figure 36. Solid rectangular body under uniaxial tension with an initial crack in the bottom middle.

The outer part of the body is modeled with FEM (blue) and the bottom center part, where the initial crack is located, is modeled with CA (red). The CA modeled area, a very fine mesh (8281 nodes), is the main point of interest because of the initial crack. The main idea in combining these two methods is to pass displacement data from FEM to CA at the interface and to supply force boundary conditions from CA to FEM at the interface in each time step. It is seen in Figure 36 that the FEM part is modeled as a rectangle with a rectangular hole in it where we filled with the CA model. When we start calculations, since the FEM modeled part does not represent the real system we are trying to model and in this configuration there is no force feedback from the CA modeled part, initially, the whole system is modeled using the FEM method ($t = 0$). After which, the originally described models are implemented. The procedure is summarized as follows:

1. At $t = 0$, model the whole system with FEM and find displacements
2. Pass the displacement data at the CA-FEM interface from FEM to CA nodes (using linear interpolation)
3. Make the CA calculations
4. Replace the FEM model with the model introduced in Figure 36
5. Calculate the strains at the interface nodes of CA model and by assuming elasticity, convert the strains to stresses and then to forces
6. Apply these calculated forces as applied forces to the interface nodes of FEM model
7. Make the FEM calculations
8. Pass the displacement data at the CA-FEM interface from FEM to CA nodes
9. Make the CA calculations
10. Go back to step 5

In theory, by using small time steps, this methodology should work. However, the lack of an ability to apply the Poisson ratio to the model has a detrimental effect on this approach. The problem starts emerging when the FEM modeled part starts contracting in the y direction under tension, but the CA modeled part cannot (Figure 37).

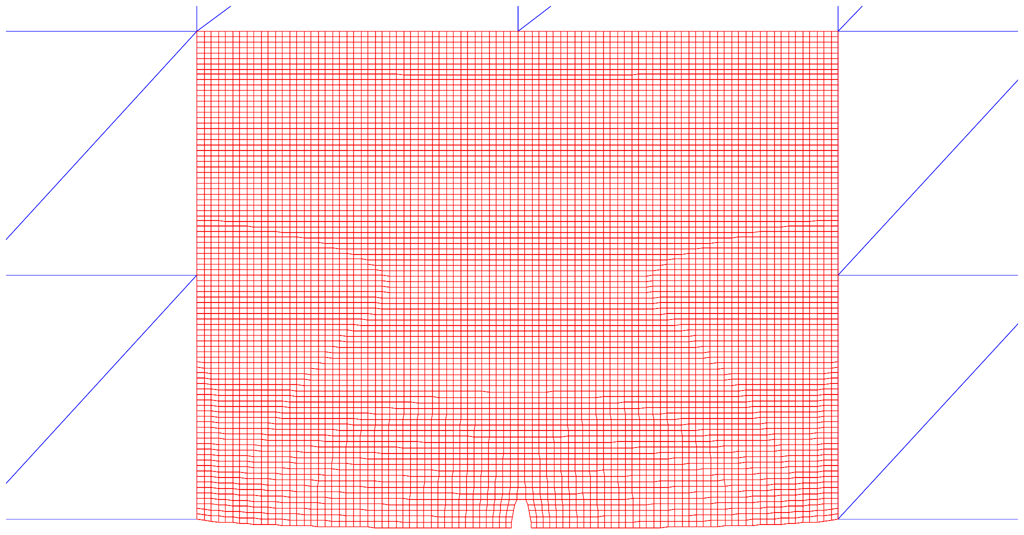


Figure 37. The CA modeled part of solid body after calculations.

It is obvious that the CA modeled part expanded from the bottom instead of contracting as the real physical system requires. The reason for this is that the CA rule always tries to bring the system to equilibrium, which is represented by the $\boldsymbol{h}$ and $\boldsymbol{u}$ spring vectors (equilibrium lengths of the springs linking the CA particles). In this case, since $\boldsymbol{u}$ is defined at the beginning when the system is in equilibrium, without any applied force, the CA model cannot compensate for the contraction of the FEM modeled part and expands.

This leads to another important problem that the stresses calculated from the CA nodes at the interface are lower than they should be because of the relaxation. These forces must hold the FEM modeled part intact, but because of this underestimation, the FEM model becomes more deformed than it should be (Figure 38). The Poisson ratio can be integrated into CA calculations by using dynamic spring vectors whose lengths can be calculated from the average contraction of the FEM nodes. This method does not ensure accurate calculation of the true strains and stresses at the interfaces, and simulations show that this method also causes unrealistic deformations on the FEM modeled part.

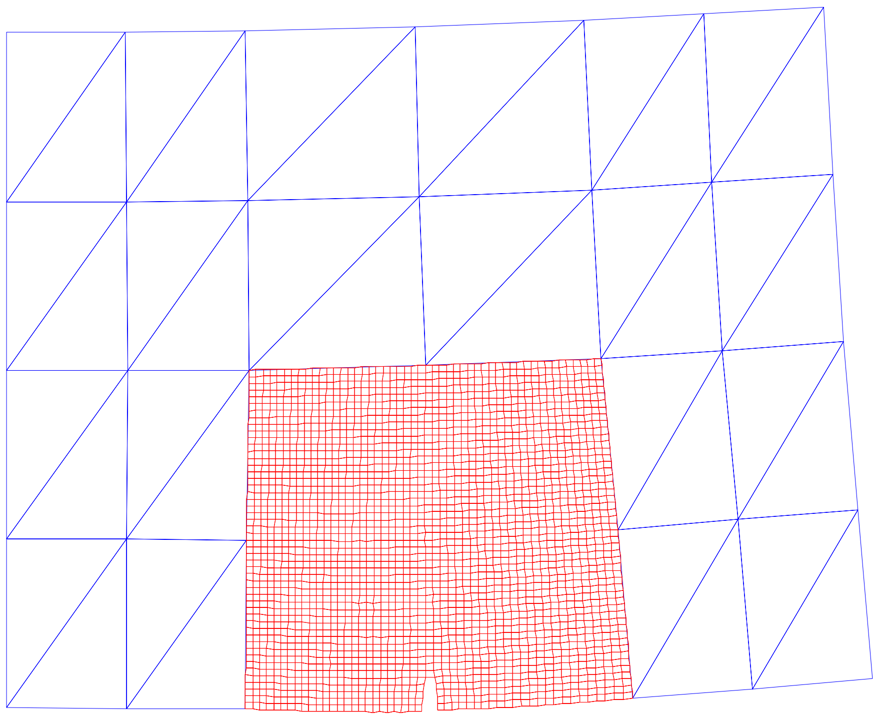


Figure 38. The excessive deformation of FEM model caused by underestimated interface stresses.

An alternate method is to use the FEM model with zero Poisson ratio to overcome the expanding of CA part. The physical model to be modeled is shown in Figure 39. In this model, instead of using force boundary conditions, we use displacement boundary conditions on the right hand side. We increase the displacements on the right hand side nodes until the first crack starts propagating, and then hold it fixed. Even though this system does not represent an actual physical system, we attempt to determine whether crack propagation can be modeled when the CA technique is coupled with the FEM technique. By using this model, crack propagation was modeled, but interestingly, changing the damping ratio γ did not significantly affect the results, as it did in the CA modeled example as a result of the sub-cycling technique. We used sub-cycling (running the CA part of the simulation several times at each time step, instead of once) to allow CA particles to come to rest at each time step. In this manner we distribute the CA particles as uniformly as possible between calculations. The propagated crack is shown in Figure 40.

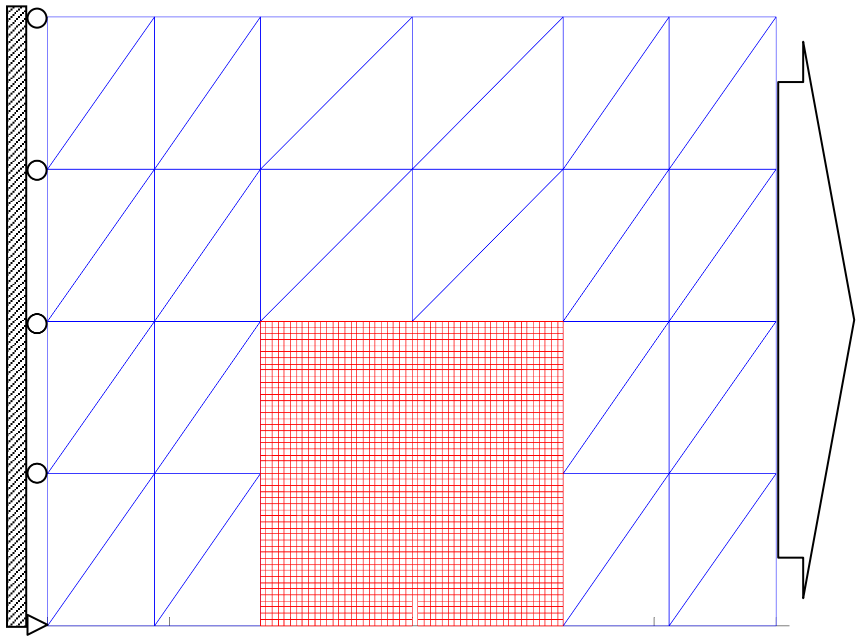


Figure 39. FEM – CA coupled model of a plate with an initial crack in the bottom middle (Poisson ratio = 0).

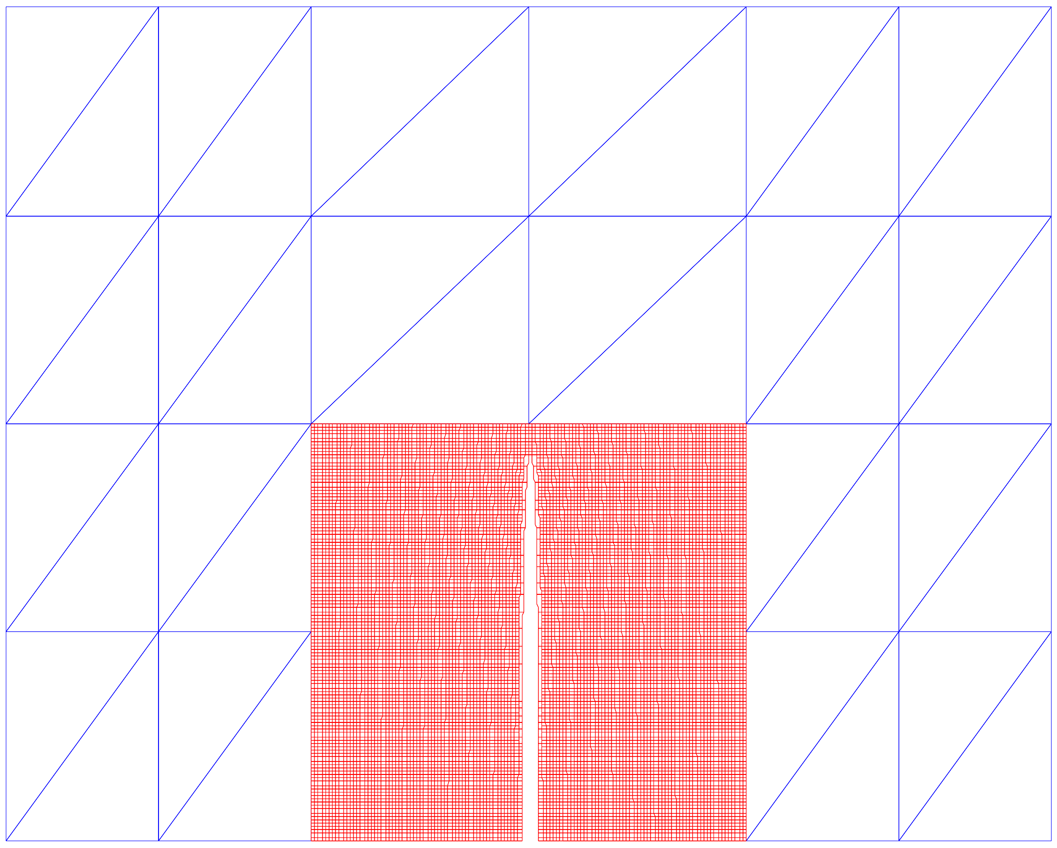


Figure 40. Crack propagation on the plate (Poisson ratio = 0).

One additional area of consideration concerns the method used to compute the forces applied at the interface. The forces that are applied to the FEM part of the model are calculated from the strains of the CA nodes at the CA – FEM interface. In implementation this leads to the problem that the forces applied at the interface are in fact the forces calculated at the previous time step, not at the current time step. This problem can be minimized by using smaller time increments, but still when run for enough time, the errors start to emerge and result with stability problems and unrealistic deformations.

These experiments and modeling efforts can be considered the first steps in enabling the development of a useful model to predict crack propagation behavior using the CA technique. Although the proposed CA rules and methodology, in its current form, cannot truly represent crack propagation on a solid body, it is important to note that it can demonstrate the behavior of the cracks at a macroscopic level by using specially adapted local rules.

VI. CONCLUSIONS

The main purpose of the Cellular Automata technique is to model the macroscopic behavior of a physical system. This modeling is performed through the use of a local rule that is implemented at the microscopic level, and is based on the premise that macroscopic behavior is the sum of microscopic movements of CA particles. In this thesis, the basic rule that was used for CA modeling was based on the reflection of a particle with respect to the geometric center of its neighboring particles.

Using this approach, we successfully modeled physical systems governed by the one and two dimensional wave equations. We also developed a method to incorporate forces into the CA models.

The most important point noted when forces were involved was the Time Scaling Factor, which yielded temporal agreement between the CA model and the real solution. In one dimensional problems, we correctly defined the TSF and also managed to couple the CA and FEM techniques. However, in two dimensional problems, we were unable to correctly define the TSF formulation (but Eq.36 is a good approximation). This explains why we were able to model problems when forces were not involved, but we could not model them when forces were involved. We found that in two dimensional systems the TSF is not only a function of physical properties of the system but also depends on the positions of the particles. This leads to a dynamic TSF, but when it comes to coupling CA and FEM models in two dimensional systems, TSF also represents the time step size for FEM calculations, which makes it harder to implement. In modeling two dimensional systems with using both CA and FEM coupled, due to the reasons explained above, we managed to model them in an error margin (5% to 30% depending on the position). For future work, TSF should be the point of interest and should be well developed to be able to model two or three dimensional systems when forces are involved.

Despite the significant benefits associated with CA modeling, there are a few major drawbacks. First, with the CA rules introduced, there is a lack of ability to implement the Poisson ratio. One way of implementing this ratio is to use dynamic spring vector lengths instead of constant ones. This can be achieved by averaging the contraction of the FEM (or any other well-developed method) elements, but this still requires another modeling method. We also tried to model fracture mechanics problems by using both the FEM and CA methods, but because of the inability to implement Poisson effect, these attempts were unsuccessful. The local CA rules are open to development and if a CA rule set that includes the Poisson effect can be developed, the CA technique may be a strong candidate for fracture mechanics problems when coupled with FEM.

The CA technique is not as standardized as FEM approaches. But, there are several powerful advantages to CA modeling. In particular, the CA approach is noted for the simplicity of its calculations and the fast computation speed, relative to FEM approaches. In CA, calculation time increases linearly with an increasing number of system degrees of freedom whereas it increases quadratically in FEM calculations.

In this thesis, we demonstrated that the CA technique is capable of modeling real physical systems governed by one and two dimensional wave equations. The CA technique can be considered a significant competitor to other numerical methods if the shortcomings discussed in this thesis are overcome.

LIST OF REFERENCES

[1] Shalizi, C., “Notes on Cellular Automata”, [http://cscs.umich.edu/~crshalizi/notebooks/cellular-automata.html], 14 June 2006.

[2] Tech rep., MIT, MAC TR-81, *Information processing and transmission in cellular automata*, Banks, E., 1971.

[3] Chopard, B., Dupuis, A., Masselot, A., Luthi, P., “Cellular Automata and Lattice Boltzmann techniques: an approach to model and simulate complex systems”, Computer Science Department, University of Geneva, 1998.

[4] Burks A., “Von Neumann’s self-reproducing automata”, *In Essays on Cellular Automata*, University of Illinois Press, pp. 3-64, 1970.

[5] Gardner M., “The fantastic combinations of John Conway’s new solitaire game of life”, *Scientific American 223*, pp.120-123, 1970.

[6] Wolfram, S., “Theory and application of cellular automata”, *World Scientific*, 1986.

[7] Wolfram, S., “Cellular automata and complexity”, *Addison-Wesley, Reading MA*, 1994.

[8] Chopard, B., “A cellular automata model of large-scale moving objects”, *J.Physics A: Math. Gen. 23*, pp.1671-1678, 1990.

[9] Margolus N. H., Ph.D. Dissertation, MIT, 1987.

[10] Wolfram, S., *A New Kind of Science*, Wolfram Media, 2002.

[11] “Cellular Automaton”, Wikipedia online encyclopedia, [http://en.wikipedia.org/wiki/Cellular_automata], 19 August 2006.

[12] Thomson, W. T., Dahleh, M. D., *Theory of Vibration with Applications*, 5th edition, Prentice Hall, 1998.

[13] Tse, F. S., Morse, I. E., Hinkle, R. T., *Mechanical Vibrations, Theory and Applications*, 2nd edition, Allyn and Bacon Inc., 1978.

[14] Kwon, Y. W., Bang, H., *The Finite Element Method Using MATLAB*, 2nd edition, CRC Press LLC, 2000.

[15] Callister, William D., Jr., *Materials Science and Engineering and Introduction*, 6th edition, John Wiley & Sons Inc., 2003.

[16] Hellan, Kåre, *Introduction to Fracture Mechanics*, McGraw Hill International Editions, 1985.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Young W. Kwon, Ph.D.
   Mechanical & Astronautical Engineering
   Code ME/Kw
   Monterey, California